**Joshua Wheeler**

**CSE 140L**

**A11362716**

**CSE140 Lab 1: Alarm Clock Behavioral Descriptions and Technical Descriptions**

**Report Overview:**

This report describes all implementation details of every component used for the functionality of the alarm clock created in LogicWorks. Part One describes how all of the behavioral diagrams were implemented. Part Two shows the updated behavioral diagrams and describes how they work. After that, the implementation details and implementation considerations are also descried.

**Part One: Basic Alarm Clock**

For this part there were 4 behavioral diagrams that needed to be implemented in order to form the core functionalities of the alarm clock.

**Behavioral Diagram One: Clock Tick Behavior**

The implementations for figure one in the lab was implemented using the clockDev Component and AlarmDev Compenents.

**ClockDev Inputs:**

CLK – Pulses the clockDev components. All other components in part II are dependent upon clockDevs clock pulse. Hooked to a clock component which sends 0 and 1 pulses to this input.

CLR – Clears Output Values to 0 for s0..5, m0..5 and h0…4. This is hooked to a binary switch to set.

Set – Enabled Mode allows SH and SM to be set, Disable mode does not allow them to be set. The set functionality is determined with the decoder component which is described below.

SH – Set the hours, any integer value between 0-59. Hooked to a push button to be manually set.

SM – Set the minutes, any integer value between 0-59 . Hooked to a push button to be manually set.

**ClockDev Outputs:**

S0…S5 – Stores values for seconds in binary.

M0...M5 – Stores values for minutes in binary.

H0…H4 – Stores values for hours in binary

The values of this component conditionally outputs to an LCD interface, behavior described in the behavioral diagram four implementation description. The Alarm Buzz behavior for this component is described in behavioral diagram two’s implementation description. The set behavior uses a decoder, described in behavioral diagram three’s implementation description.

**The AlarmDev Component** Tick Behavior is the same as clockDev except for the following differences:

* It does not need a CLK input since all values in this component are manually set.
* It does not track seconds

**Behavioral Diagram Two: Alarm Buzz Behavior**

The implementation for figure two of this lab was achieved using three 6Comp components, one for each hours, minutes and seconds respectively. The outputs were directed to an and gate equality checker which determines whether or not the buzzer is on.

**The 6Comp Component**

Inputs were the hours minutes and seconds for the alarm clock. Each of the three 6comp components stored one of each respective quantity for the alarm and the clock. The output of this component is 0 if the input values are not equal form the alarm and the clock and 1 otherwise. The outputs are hooked to an and gate used as an equality checker.

**The Equality Checker**

If all values input to the and gate are one, it will output on and turn on the buzzer

**The Buzzer** Has an one input which determined whether it is on (value 1) or off (value 0). Will Buzz if on, does not buzz if off.

**Behavioral Diagram Three: Time and Alarm Set Behavior**

The time and alarm set behavior was implemented using a decoder, which determined one of three states. Alarm set enabled, clock set enabled or neither.

**The Decoder:**

Inputs:

S0 – Inputs 0 or one. Determines alarm set.

S1 – Inputs 0 or one. Determines clock set.

EN – Whether or not component is enabled.

Outputs:

Q0 – Not hooked up, unneeded. Indicates set behavior on neither enabled.

Q1 – If 1, this means s1 is one which enables set behavior on clockdev

Q2 – If 1, this means s0 is one which enables set behavior on alarmdev

Q3 – Not hooked up, unneeded. Indicates set behavior on both are enabled, cannot set both at same time.

**Behavioral Diagram Four: Alarm Clock Structural Schematic**

The structural schematiclargely determines how the various components communicate to each other which has already been defined in the component descriptions above. These interconnections ultimately determine the values on the LCD interfaces. The below components describe how values are output to the various seven segment displays.

**Mux 2x6:**

These conditionally output the binary values of the clock dev and alarm dev quantities for minutes and hours. There are a total of two muxes for part one, one controls the minutes, the other controls hours.

Inputs:

B0…B5 – This stores the bits for the alarm values for minutes or hours respectively for each mux.

A0…A5 – This stores the bits for the clock values for minutes or hours respectively for each mux

S – Determines whether B0…B5 or A0…A5 becomes the output for Q0…Q5 which connects to the LCD Interface.

Outputs:

Q0…Q5 – This outputs the binary values on the bus for either A0…A5 or B0…B5 which may contain minutes or hours from clockdev or alarm dev respectively.

**LCD Interface:**

This component takes the output of the Mux as input and outputs the binary values to 7 segment displays which display the value on the clock.

Inputs:

0…5 – The bits which store hours, minutes or seconds respectively. The seconds are not hooked to a MUX.

Outputs:

A1…g1, a2…g2 – These outputs connect to 7 segment displays to show the number on the clock

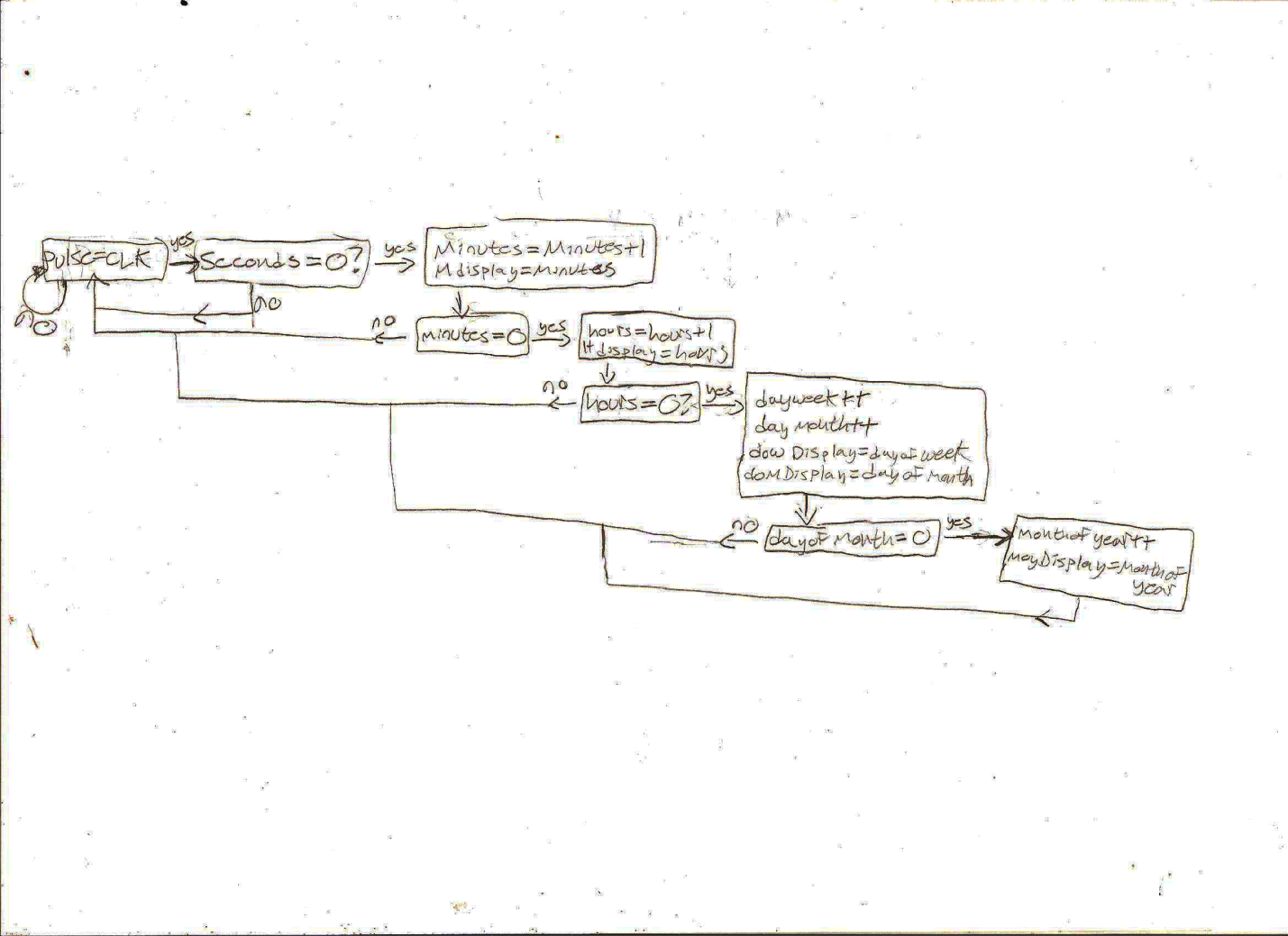
**Seven Segment Display:**

This component displays the decimal integer output visually from its inputs which store the value to be displayed.

**Part Two: Days and Months**

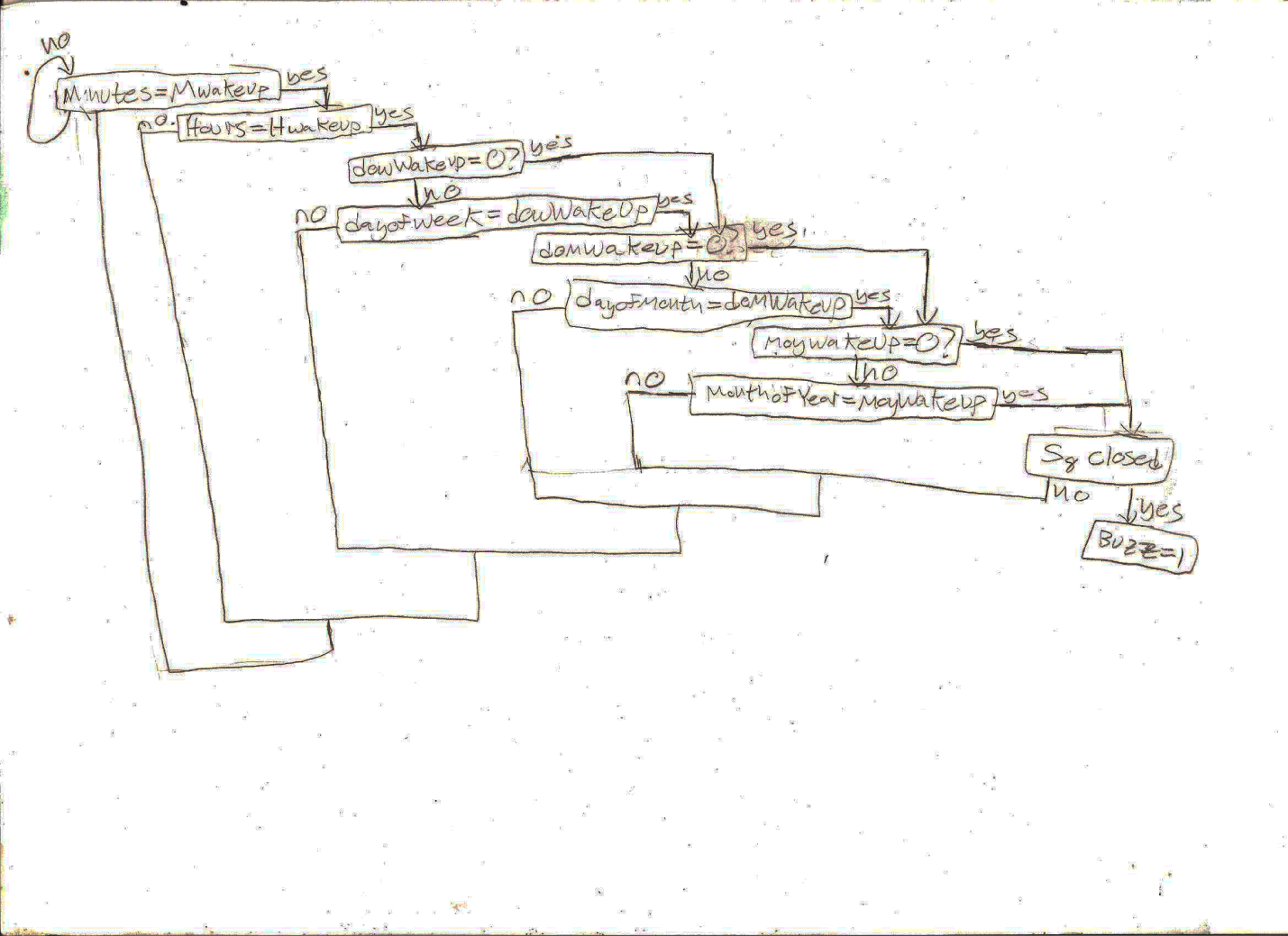
Part to consisted of adding additional functionality to display the day of the week, day of the month and month of the year for the alarm clock and also integrating it into the alarm buzz functionality. Behavior diagrams and the inputs and outputs of all 6 components are described below with their implementation considerations.

**Diagram One: Part II Clock Tick Behavior**



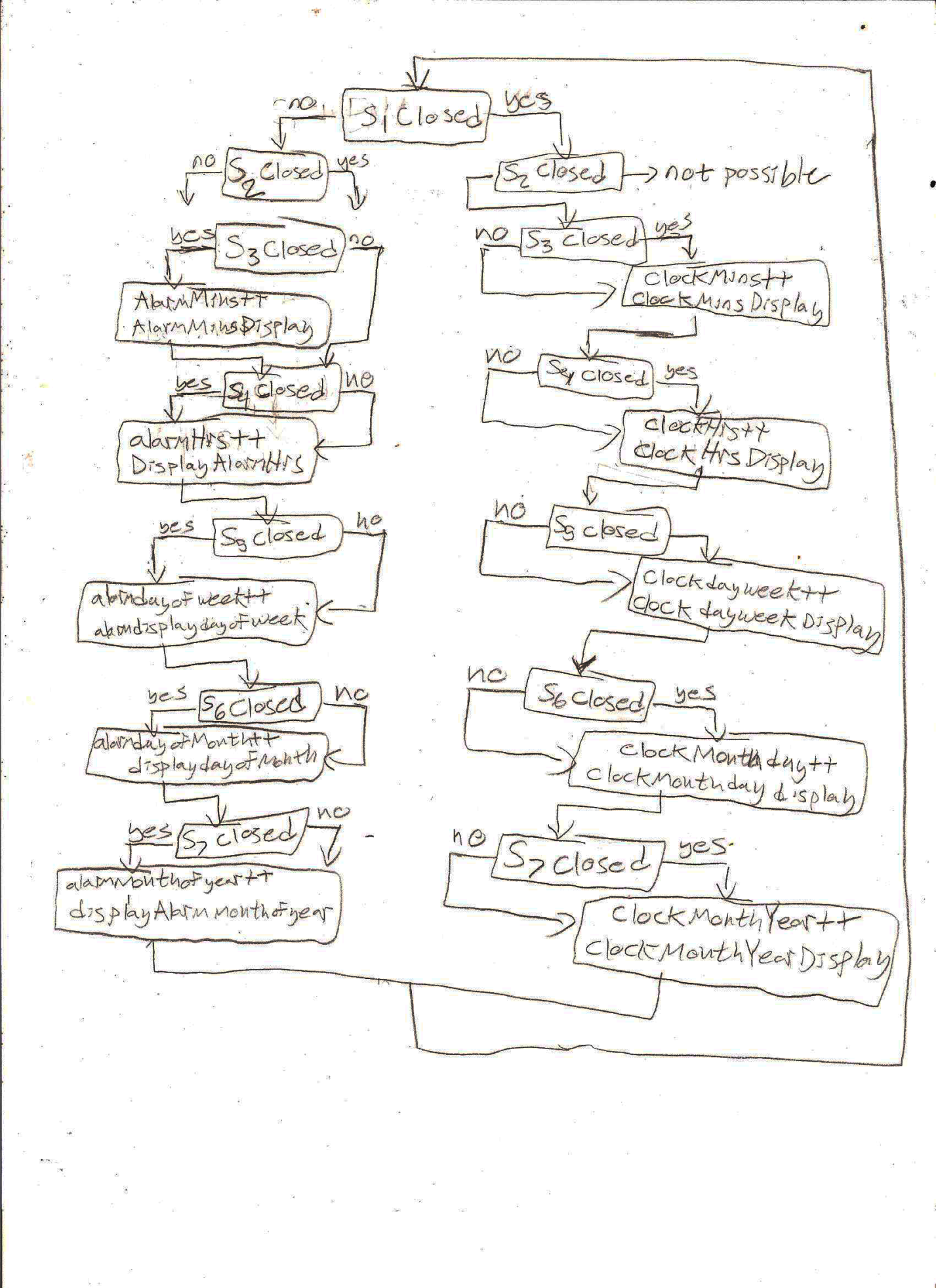
The tick behavior in part two was modified increment the day of week, day of month and month of year. If hours are zero, day of week and day of month will increment at the same time, if day of month is equal to zero, month of year will increment. If any condition is false, the clock will send a pulse to seconds.

**Diagram Two: Part II Alarm Buzz Behavior**

****

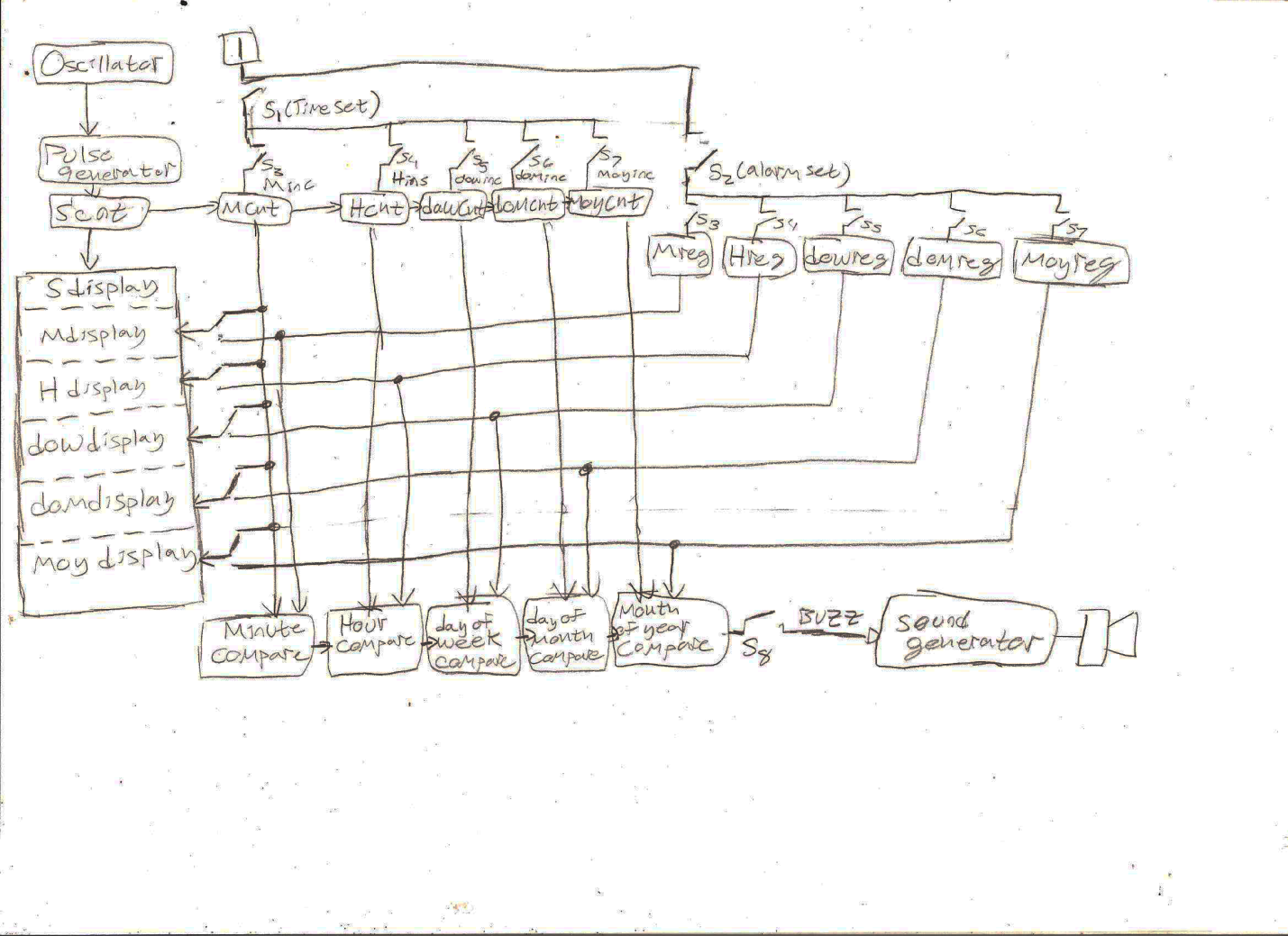
The modified alarm buzz behavior for this part is similar to that of part one for the three new quantities. The only modification made to the original behavior diagram for the buzz behavior is that for day of week, day of month and month of year a check for value zero on the alarm is done before a check for equality to increment the respective quantity. If the zero check is true, then the equality check for the quantity with value zero is ignored.

**Diagram Three: Part II Time and Alarm Set Behavior**

****

The time and alarm set behavior is nearly exactly the same as the part one set behavior, except that day of week, day of month and month of year have been added on with their own respective switches.

**Diagram Three: Part II Structural Design**

****

The structural design of part II is very similar to that of part one. The only differences are with respect to the day of week, day of month and month of year being added on to the display and the comparator chain.

**dowClock Component:**

The day of the week clock uses a counter-4 component to set its values. The goes from 1 to 7. Once the counter reaches the binary value of 1000 (eight), it will reset to one by outputting the 1 bit from Q3 output into the LD input.

Inputs:

Set – When set is on, the set week may pulse the clock

SW – The is the set week input to the day of week component. When set to one, the value increments.

CLK – when the clock detects a pulse, the counter is incremented.

UP – This indicates to the counter that it should count up.

CLR – This clears all the output Q0…Q3 bits to zero.

D3…D0 – These set the initial values of the outputs.

LD – This is the load input, the load input is hooked to the output of Q3, it loads a one when Q3 is set to one.

EN – This is the Enable input, having it set to on enables the counter tick.

Outputs:

Q0…Q3 – Each output stores a 0 or 1 bit representing the value of the day of the week. When Q3 is 1, the counter resets.

CO – The carry-out output is not needed.

**dowAlarm Component:**

This component is implemented exactly the same as the dowClock component. The only difference is that Q3 is not used, and LD loads a zero.

**domClock Component:**

This component is primarily controlled by a counter-8. The Q6 and Q7 outputs are not used, only the first 6 output bits are needed. This component element takes the month of the year bit values as inputs in order to conditionally determine how high the counter can count before it resets. The implementation for this behavior is described for each respective input and output for all of the subcomponents of the domClock component

Inputs:

M0…M3 –These inputs are set by the month or the year clock component,

CLR – Sets all output Q0…Q7 bits to zero

Set – Determined if output values can be incremented

SD – This is connected to an and gate, if both set and Set Day are on, the clock will pulse.

CLK – When this is pulsed, the counter increments.

UP – This indicated that the counter should count up.

D0…D7 – These determine the initial outputs of the counter, they are all grounded.

LD – The Load input is connected to a reset line. The reset line output is dependent upon a series of conditional arguments which determine at which value the counter should reset at. The way this conditional logic is implemented is described by the conditional counter behavior below.

EN – This determined whether or not the circuit is enabled.

Outputs:

Q0…Q7 – These output bits represent the value of the day of the month. All of the output lines are also connected to 3 different and gates, an 8-and and two 5-and’s. The 8-and outputs a 1 when the output is 32, one 5-and outputs a 1 when the output is 31 and the other 5-and outputs a 1 when the output is 29. These and gates help to determine the conditional counting behavior. The component will reset at the counter value for whichever AND gate outputs a one.

Conditional Counting Behavior:

The counting behavior of this circuit element is controlled by the M0…M3 inputs are hooked to 12 different buses that each connect to an AND gate. This AND gate outputs a one corresponding to the value of the month it is assigned to. Each month value corresponds to 30, 31 or 28 days. Each AND gate tracking the value of the month is hooked to an OR gate for the respective amount of 30, 31 or 28 days allotted to that months value. The OR gate will output a one to another AND gate which also hooked to the output of the domClock Component. If the domClock Component output resets at 28 and the M0…M3 input is a 2, then this and gate will set a reset line which will reset the domComponent to a one.

**domAlarm Component:**

The domAlarm component is exactly the same as the domClock component except for the following differences:

-It does not conditionally reset depending on the day of the month, so the gates described in the conditional counting behavior above do not exist

-The domAlarm component always counts between 0 and 31

-This needs no M0…M4 inputs as there is no conditional logic needed to increment the counter.

-Once the domAlarm reaches the value output value of 32 it resets to 0.

**moyClock Component:**

The month of the year clock is implemented exactly like the day of the week counter. It uses a counter-4 and resets once the value of the counter 4 reaches 13, the value 13 is detected by an AND gate that is connected to the output lines. The AND gate resets the initial value of the counter to 1 by being hooked to the LD input.

**moyAlarm Component:**

The month of the year alarm component is implemented exactly like the moyClock component. The only difference is that the AND gate connected to the output lines of the counter-4 outputs the the CLR input of the counter rather than the LD input.

**Implementation Considerations for the Structural Design of Part II:**

There were some logic designs in the structural design not yet mentioned above. The first is how the output of the hours increments the day of the week, and how that chains to incrementing the day of the month and month of the year. In order to do this, an AND gate is connected to the output lines of the H0…H4 bus of the clockDev component which outputs a one when the input pattern to the and gate is 00000 which indicates the hours are zero. The output of this and gate is connected to the set inputs of both the day of the week and day of the month counters which are incremented at the same time. The output of the day or the month connects to the set input of the month of the year counter to increment the month of the year once the day of the month rolls over to 1.

The next implementation consideration to describe is how the comparator works after the part II implementation. This works the same as the comparator functionality for part I except that the output of the 6comp is connected to an OR gate. The OR gate acts as a conditional check which outputs true if either the equality check passes or the value of the measured quantity on the respective alarm component is zero. The outputs for each of these conditional checks are hooked to an AND gate which acts as the equality checker as implemented in part one. If all of the quantities for minutes, hours, day or week, day of month or month of year are equal, this and gate will output a one. An alarm clock value of zero is ignored due to the conditional check with the OR gate. In order to do things like compare 5 bit signals with 6 bit comparators, the unused input lines were grounded which was processes as being equivalent to the zero bit in LogicWorks.