EE 552
Final Report:
Integrated Automotive Diagnostic System

Designed for –
Dr. D. Elliott

Designed by –
Patrick Chan - 364305
Neil Fraser - 252885
Srilata Kammila - 253698
Edmund Quan - 244667

November 23, 1998

Table of Contents

Abstract

In the automobile industry, as the need for feedback and security systems in a vehicle increases, it becomes essential to centralize and efficiently manage information from these various systems. Moreover, since the electronic components are becoming more inexpensive, implementing a centralized digital system is now more feasible than ever before. Therefore, as a part of our project for EE 552, we have implemented a diagnostic and feedback system for vehicles via a centralized digital system.

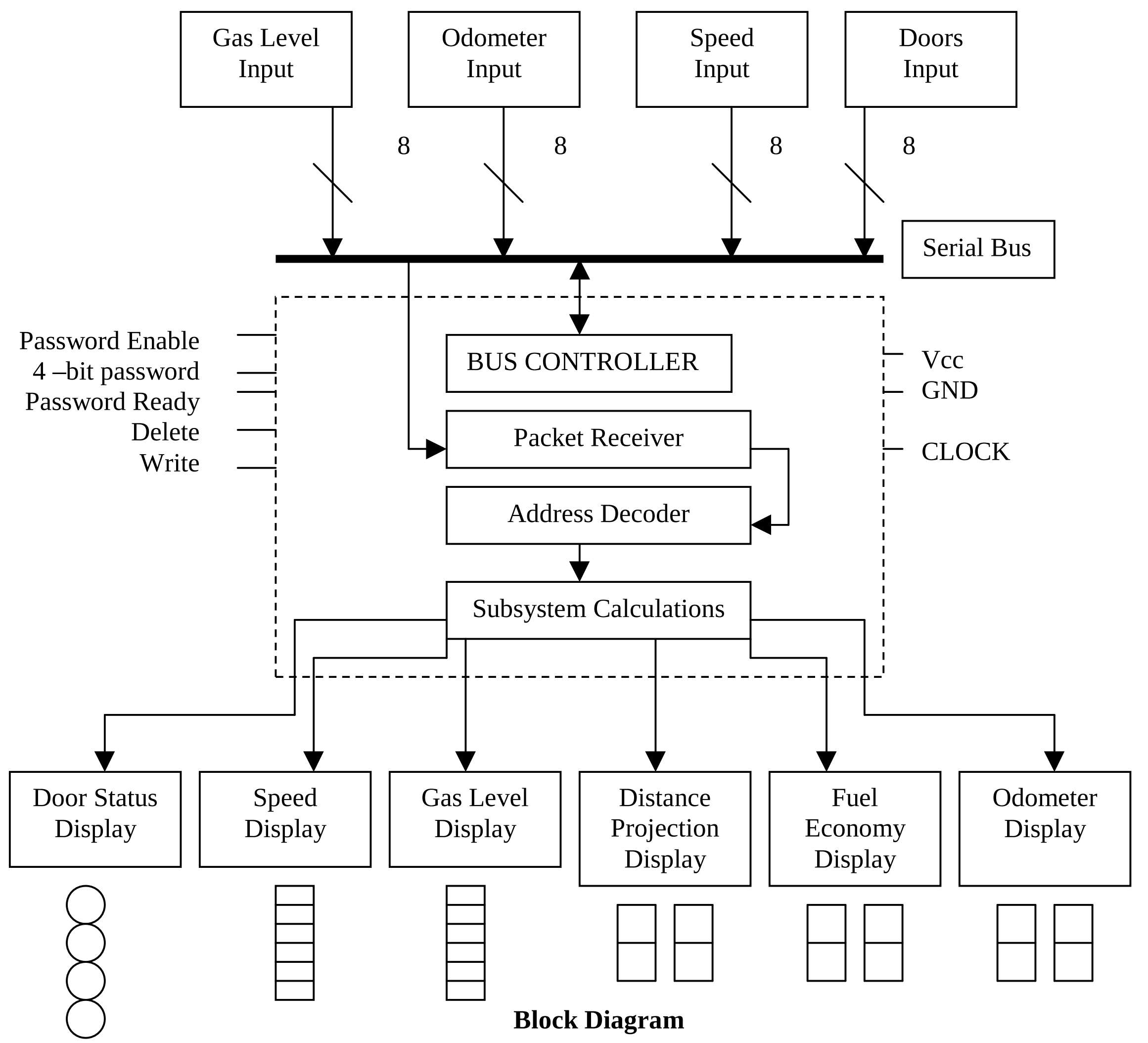
The design of the system has been carried out by sub-dividing the system into various subsystems. These include the Gas Level System, Vehicle Speed System, Distance Projection System, Fuel Economy System and Door Monitoring System. These various systems communicate via a single bus line, which is controlled by a Bus Controller System. The design of the entire system has been accomplished successfully.

The design was tested by simulating the system as a whole using Max+PlusII CAD Tools. The CAD simulations have yielded the desired results. The design was uploaded to a FPGA and tested. The FPGA simulations have also yielded the desired results. The system draws its inputs from components representing Gas Level, Vehicle Speed, Distance Traveled and Door Status. It is capable of communicating via a common bus. The system displays Door Status, Gas Level and Vehicle Speed and performs the appropriate calculations to output the Distance Projection and Fuel Economy. The system successfully implements our Integrated Diagnostic Feedback System.

IC Data Sheet
IADFS98 – Integrated Automotive Diagnostic and Feedback System

General Description
The IADFS98 is a digital data acquisition and monitoring system capable of providing feedback on (currently) four 8-bit inputs representing common components found in a vehicle. These 8-bit inputs are received onto a single bus, which provides centralization and integration of the entire system.

Features
- Easy interfacing with serial inputs through a common bus
- 8-bit binary inputs
- External open collector serial bus
- 8-bit binary outputs
- Typical clock frequency 1.2 kHz



Block Diagram

Functional Description

The device takes four 8-bit inputs that are serially put on a common bus. The addresses of these are hard-coded and a user wishing to monitor any feature provided by the device needs to plug in the input in the appropriate slot. The inputs are chosen on polling that is done by the bus. The addresses of the input devices are compared with the address currently on the bus and the open collector is pulled down when an input is presented by the input device. The inputs are transmitted by the bus and received by the appropriate data manipulation modules. The system uses a clock divider that operates at the frequency 1KHz. The device provides 8-bit binary outputs.

Specifications

- Internal clock of the FPGA is at 25.125 MHz . The system uses a clock divider that operates at a frequency of 1KHz. The maximum speed of the system is 1.2Khz.
- The logic cells count utilizing one board is 1096 blocks. Two boards are used to implement the system for demonstration purposes. The boards would correspond to the input and output sides utilizing 950 and 768 logic blocks respectively.

Label	**Flex10K (pin #)**	**UP 1 Board (hole #)**
Clock	91	Clk
Reset	54	EXPA - 22
Password enable	41	Flex Switch 1
Password write	40	EXPA - 2
Password ready	39	EXPA - 3
Password delete	38	EXPA - 4
Password error	101	EXPA - 56
Password match	100	EXPA - 55
Password in (3)	36	Flex Switch 5
Password in (2)	35	Flex Switch 6
Password in (1)	34	Flex Switch 7
Password in (0)	33	Flex Switch 8
Bus_com_port	98	EXPA - 53
Bus_recv_port	95	EXPA - 51
A/D_clock	50	EXPA - 19
A/D_pulse	51	EXPA - 20
Odometer_input(0)	53	EXPA - 21
Odometer_input(1)	55	EXPA - 23
Odometer_input(2)	61	EXPA - 25
Odometer_input(3)	63	EXPA - 27
Odometer_input(4)	65	EXPA - 29
Odometer_input(5)	67	EXPA - 31
Odometer_input(6)	70	EXPA - 33
Odometer_input(7)	72	EXPA - 35
Speed_input(0)	76	EXPA - 39

Speed_input(1)	79	EXPA - 41
Speed_input(2)	81	EXPA - 43
Speed_input(3)	83	EXPA - 45
Speed_input(4)	86	EXPA - 47
Speed_input(5)	88	EXPA - 49
Speed_input(6)	95	EXPA - 51
Speed_input(7)	98	EXPA - 53
Gas_input(0)	207	EXPC - 39
Gas_input(1)	204	EXPC - 37
Gas_input(2)	202	EXPC - 35
Gas_input(3)	200	EXPC - 33
Gas_input(4)	198	EXPC - 31
Gas_input(5)	195	EXPC - 29
Gas_input(6)	193	EXPC - 27
Gas_input(7)	191	EXPC - 25
Door_input(0)	239	EXPC - 55
Door_input(1)	228	EXPC - 53
Door_input(2)	226	EXPC - 51
Door_input(3)	223	EXPC - 49
Door_input(4)	221	EXPC - 47
Door_input(5)	219	EXPC - 45
Door_input(6)	217	EXPC - 43
Door_input(7)	214	EXPC - 41
Serial_bus(0) : Odometer	45	EXPA - 15
Serial_bus(1) : Gas	46	EXPA - 16
Serial_bus(2) : Door	48	EXPA - 17
Serial_bus(3) : Speed	49	EXPA - 18
Gas_output(0)	162	EXPB - 55
Gas_output(1)	159	EXPB - 53
Gas_output(2)	157	EXPB - 51
Gas_output(3)	154	EXPB - 49
Gas_output(4)	159	EXPB - 47
Speed_output(0)	149	EXPB - 45
Speed_output(1)	147	EXPB - 43
Speed_output(2)	144	EXPB - 41
Speed_output(3)	142	EXPB - 39
Speed_output(4)	139	EXPB - 37
Door_output(0)	137	EXPB - 35
Door_output(1)	134	EXPB - 33
Door_output(2)	132	EXPB - 31
Door_output(3)	129	EXPB - 29
Projection_output(0)	175	EXPC - 15
Projection_output(1)	182	EXPC - 17
Projection_output(2)	184	EXPC - 19
Projection_output(3)	186	EXPC - 21

Projection_output(4)	188	EXPC - 23
Projection_output(5)	191	EXPC - 25
Odometer_output(0)	193	EXPC - 27
Odometer_output(1)	195	EXPC - 29
Odometer_output(2)	198	EXPC - 31
Odometer_output(3)	200	EXPC - 33
Odometer_output(4)	202	EXPC - 35
Odometer_output(5)	204	EXPC - 37
Fuel_economy_output(0)	214	EXPC - 41
Fuel_economy_output(1)	217	EXPC - 43
Fuel_economy_output(2)	219	EXPC - 45
Fuel_economy_output(3)	221	EXPC - 47
Fuel_economy_output(4)	223	EXPC - 49
Fuel_economy_output(5)	224	EXPC - 51

EXPA – Flex Expansion A
EXPB – Flex Expansion B
EXPC – Flex Expansion C

Chip Overview

The chip accepts four 8-bit inputs that are transmitted onto a common open collector bus. The 8-bit inputs are a result of converting analog inputs (voltages) into digital inputs using an Analog-to-Digital converter prior to being put on the bus. Each address is hard-coded into each input system, a developer wishing to monitor any feature provided by the device must attach the input and corresponding output to the appropriate pins. The inputs are chosen on polling, which is done by the bus. The addresses of the input devices are compared with the address currently on the bus and the open collector is pulled down when an input is presented by the input device. The inputs are transmitted onto the bus by the corresponding packet transmitter which shifts out the input bits. These are received by a common receiver which loads the data in parallel in to a set of registers. The addresses are decoded and the inputs sent to the appropriate system modules. The system then generates binary outputs after the necessary data manipulation has been performed. For demonstration purposes these binary outputs are then converted to be displayed on LEDs. The system uses a clock divider that operates at the frequency 1KHz.

IC Testing

Before the actual system was tested, the individual input and output hardware components were tested using a protoboard. The A/D converters were given a range of voltages from 0 to 5 Volts. The binary outputs were analyzed and confirmed to have provided accurate results. For the purpose of testing, the components forming the output system the above binary outputs were fed into Binary-to-BCD and BCD-to-LED converters. The correspondence between the supplied voltages and the final LED displays is as follows:

Voltage (Volts)	7-Segment LCD Display
0	0
0.5	6
1	13
1.5	19
2	26
2.5	32
3	38
3.5	45
4	51
4.5	58
5	63

After the above testing was conducted to ensure proper operation of the hardware components, further testing was done directly on the prototype system.

The inputs to the prototype system were provided as varying voltages representing Gas, Distance and Speed of the vehicle, another set of inputs were provided by switches representing the status of the four Doors. The analog inputs ranging from 0 to 5 Volts were fed into A/D Converters. The digital outputs from all four components were directed to input modules within the FPGA. The Bus Controller then polled the modules and these modules then put the data onto the bus serially. This functionality was tested by using a voltmeter to monitor the value on the bus. To allow this testing, we used a slower clock to allow us to observe the information being put on the open collector bus. By disconnecting the Bus Controller Line and observing if the value on the bus become high, we could detected if the line was pulling the bus low. The data being transmitted was observed by using LEDs that displayed the values of the bits being shifted out by the transmitter. Meanwhile, the bus was put in high impedance so that no additional inputs would be put onto the bus. This functionality was tested by using an LED that showed when the bus was in the hold state.

On the receiver side, the LED indicators were used to display the values that were being shifted into the receiver to test if these were identical to the data bits being put onto the bus. Parallel loading of these bits into registers was tested via a sequence of LEDs. The output observed here was identical to the input put on the bus.

The inputting, transmitting and receiving of the data has been tested and confirmed to be accurate at this point in time. Final testing remains to be performed on the modules that process data from the receivers and display the appropriate results.

Design Details

Problems and Solutions

During the course of the development of the project we ran into problems with the modules. Here are some specific problems and solutions.

Fuel Economy

The fuel economy module used a division component which gave us a quotient and a remainder. We wanted to multiply this remainder by 100 and divide again to get the decimal but this would have caused us problems. The number of unused logic cells on the FPGA was very low and adding another multiplier would have made the project too large. Instead of just ignoring the remainder we rounded either up or down. This gave us a descent compromise between accuracy and efficient use of the logic cells.

Distance Projection

The distance projection was calculated from the remaining fuel in the tank and the fuel economy. These two numbers are 8-bits long thus a 16-bit product is obtained. An average car could travel a maximum distance of less than 512 km on one tank of gas which is 9-bits. Our output hardware however is only 6-bits so obviously we cannot output a number larger than 63 km. The binary to bcd converters took a 6-bit input thus our limitation. We can demonstrate that our project works by having a distance projection of less than 63. Adding another 7 segment display would have increased the amount of hardware required and the amount of time required to wire it.

Odometer

The odometer runs into the same resolution problems as the distance projection. The largest number that can be displayed on the odometer is 63km. The project still functions correctly with this limitation and can be easily modified to correctly display thousands of km. Note that an average odometer can display up to 999 999 km. This would require 20 bits of input, 6 seven segments displays, and binary to bcd converters of 20 bits.

Speed

Speed levels are outputted on a bar graph type display. This display works in the following way. Each level represents one bit of the speed. Scanning from the MSB to the LSB, when the first 1 is encountered, the graph display will light up all of the LED’s from that bit down to the LSB. This means that we can see changes in the MSB in speed. This is hardly ideal for a real car since you would only know that you are driving somewhere in a range of speeds. Ideally that range should be 1km but our range varies and is large. Speed is not used in any calculations and it needs only be seen, its value is not important for demonstration purposes.

Gas Level

The gas level is outputted on a bar graph type display. Its output works in the same manner as the speed output. The gas level on most cars is displayed as a level as we display it. It's value may be difficult to read off of the display but its value is accurately held for calculations with in the FPGA.

Packet Transmitter and Packet Receiver

The main problem that we encountered with these two modules was the synchronization of the two. We needed to ensure that the transmitted data was received properly. To do this we used a start bit and received the data by sampling in the middle of its data bit. This way we were sure not to sample at the rising or falling edge of the bit which might cause erroneous data.

Serial Bus

The serial bus is a very important part of the project. It delivers data to the modules that need it in order to make calculations and output the data. Since the serial bus is going to have multiple modules reading it and writing to it, we have to insure that the serial bus does not have more that one data value being pushed onto it a time. The solution was to use a serial bus with an open collector design. This means that the bus is always high until it receives a 1 which will drive it low.

Bus Controller

Originally we had wanted the input modules to send data when they wanted to. This would mean that if the gas system had data to send then it would request the bus and send the data. This method was asynchronous and caused us a lot of problems. We would have had to design an arbitrator system to deal with two or more inputs requesting the bus at the same time. This would have meant having priorities and would have added substantial overhead. The project doesn't fit on one chip as is anyway. To solve this problem we resorted to polling. Now the bus controller simply polls each device to send data.

Additional Considerations

As an example of estimating the number of logic cells used consider the following data from our project.

Individual modules before integration
IADS – 300 logic cells
Fuel Economy – 180
Distance Projection – 181
Password – 100
Gas Level – 7
Door Status – 4
Total – 772 logic cells.

Before integrating our components we used 772 logic cells. Therefore we thought that we would have plenty of room on the chip but after integrating all of the components the total number of gates went up 296 logic cells to 1068. This meant that we were just about out of room on the FPGA. Integrated the modules can consume a lot of logic cells and a high estimation for integration should be included.

Design Verification (Simulation)

The design was verified using Max+Plus II CAD Tools to simulate inputs and generate output. The simulations are attached in the Appendix. The following simulations have resulted in the desired results:

- **IADS Simulation:** The simulation resulted in various subsystem input values observed propagating through the systems. The Bus Controller Simulation was done as part of the IADS simulation and not individually. It polls and chooses the inputs to be put on the bus. The Automobile Speed simulation was done indirectly through the IADS since it does not have a distinct module. The simulation resulted in the value being stored in a register, which was then viewed.

- **Access Password Simulation:** The simulation was successful in detecting a valid password. There are two outputs for this system. One indicates a match, the other indicates an error.

- **Input Monitor Simulation:** The inputs from the individual systems were successfully converted into data-packets after the required change in input was observed. The request for transmission was enabled after the packet had been formed.

- **Packet Transmitter Simulation**: The shift register that forms this functional unit produced the desired results. When enabled it shifted the serial bits and loaded the data on to the bus.

- **Gas Level Simulation:** The gas input value was interpreted by the module and the corresponding level indicator was set high. Depending on the range of gas values observed on the input, one of the three levels is set high.

- **Distance Projection Simulation**: The gas level and fuel economy values stored in the respective registers were used to calculate the distance. The module successfully calculated the distance that could be covered on the remaining fuel. The output is in binary.

- **Fuel Economy Simulation:** The gas level and distance values stored in the registers were used to calculate the fuel economy value. Currently, the simulation does not produce decimal resolution. Also, the output is left as a binary value.

- **Door System Simulation:** The status found in the data packet is output to the LED's. A one in the packet implies that the door is closed and output is zero to indicate that the light should be off. A zero in the packet is interpreted as door open and output is a one to indicate that the light should go on.

The simulation of the entire chip is also completed. This includes the simulation of the IADS file which integrates all the modules mentioned above. The automobile speed, which does not have a distinct module, is not integrated yet. The simulation shows the flow of data from the input modules to the output components via a serial bus. The Gas System, Distance, and Door Status input vectors are read in. The data and the addresses from these are combined into a packet that is transmitted via the bus to a single receiver. The appropriate modules receive the data and the outputs are set in the corresponding modules. The simulation of the entire chip is attached in the appendix along with the individual simulations.

Design Description and Module Interfacing

The design of the current feature set implemented is detailed in the following paragraphs.

IADS Module **-** This is the highest-level design component that instantiates the other components that form the sub-ordinate modules including the Input Monitor, Bus Controller, Packet Transmitter, Packet Receiver and the individual system modules. The IADS also contains a Access Password Module that controls the enabling of the other components.

Access Password Module **-** The input provided to this module will contain a numeric password value. This value is compared with a predefined password and permits or denies the entry into the vehicle.

Input Monitor Module **-** An instantiation of this module corresponds to a specific input system such as the Gas Level System, Door System etc. The module detects a change in input values and supplies the data-packet to a Packet Transmitter.

Packet Transmitter Module - This is essentially a shift-register that when enabled converts the parallel data received from the Input Monitor into serial bits. The bits are transmitted on the open collector bus to a Packet Receiver.

Bus Controller Module **-** Each input monitor has a unique address which is compared with the address currently on the bus. The input is acquired if the address matches. In this way the bus polls the devices cyclically. Meanwhile, it waits until the Packet Transmitter has finished transmitting data before it resumes polling. In addition, it also enables or disables the Packet Receiver from loading the data. This ensures that only valid data is received.

Packet Receiver Module **-** This is also a shift register that, when enabled by the Bus Controller, converts the serial data into parallel data and loads the data into registers.

Address Decoder Module **-** This is contained as a process inside the IADS module. It functions as a decoder that selects the address of the received packet and maps the data packets onto the appropriate system modules.

Gas Level Module - The gas level observed is provided as an input to this module. The input is converted into a format that would be displayed using 5 LEDs. Depending on how much fuel is remaining, “low”, “medium-low”, “medium”, “medium-high” and “high” are indicated using a LED bar display.

Automobile Speed Module **-** The observed speed is input to this module. This is converted to the appropriate format that would be displayed on LEDs. Depending on how fast the vehicle is moving, “low”, “medium-low”, “medium”, “medium-high” and “high” are indicated using an LED bar display. This display format was used due to limitations in available space on the FPGA. This module also conveys information to other modules that require the speed to make distance projections.

Door Status Module **-** The input provided to this module contains information about each door being open or closed. This status is indicated by an LED display. The status of all four doors is implemented.

Distance Projection Module **-** This module acquires inputs from the Gas Level Module, Automobile Speed Module and an additional input of distance. The inputs are processed to estimate the distance that could be traveled given the current measurements. For every change that occurs in either of the inputs, the projection is calculated by multiplying fuel economy and distance. The output is implemented as binary values.

Fuel Economy Module **-** This module acquires input from the Gas Level Module and an additional input of distance. The fuel consumed per distance traveled is computed and the output is displayed in binary form.

Test Cases

IADS:

The IADS was tested as a whole incorporating the sub-ordinate module's testing. Thus this simulation also accomplishes integrated system testing.
The test cases were:

1. 0000000000000000
2. 0000000011111111
3. 1111111100000000
4. 1111111111111111

The cases were boundary values, testing the extremities of the inputs. The 16-bit vector represents the two 8-bit inputs from two input systems. These could be the outputs from the A/D converters. These were split and concatenated with their respective addresses by

the Input Monitor module. One of these is chosen by the Bus Controller to be put on the bus. The Packet Transmitter shifts out the bits of the chosen data-address packet.
The waveforms for the 3rd case listed above are attached in the appendix.

Password Access Module:
The module can be used to set, change or delete a password. The module was tested by setting the passwords as:

1. 13 - 1101
2. 10 - 1010
3. 15 - 1111

Outputs show the passwords set.
The validity was tested and appropriate output was observed
The deletion of password was also testes and the password '10' was deleted.

Gas Level Module:
Gas system was tested using values of gas that represent low, medium and high levels. A value in the range $0 \leq$ value < 8 litres set the low level indicator to high. A value in the range $8 \leq$ value < 32 set the medium level indicator to high. A value greater than 32 set the high level indicator to one. The waveform is attached in the Appendix. The test cases were chosen to represent boundary values and an intermediate value. The list of test cases is as follows:

1. 0
2. 7
3. 8
4. 32
5. 45

Door Status System Module:
The input was chosen to test a change in the status of the doors. Two of the doors were chosen to close or open simultaneously. The most significant four bits are insignificant for this system. The test cases are:

1. XXXX0101
2. XXXX1010
3. XXXX0111
4. XXXX0000

Projection Module:
Inputs are gas level (8-bits), and fuel economy (8-bits). Output is a 16-bit vector.

Test Case	Gas Level	Fuel Economy	Projection	Correct?
1. Boundary Check	11111111	11111111	1111111000000001	Yes
2. Boundary Check	00000000	11111111	0000000000000000	Yes
3. Random Calc.	00001111	00001010	0000000010010110	Yes
4. Random Calc.	10000000	10101101	0101011010000000	Yes

Fuel Economy Module:
Inputs are gas level (8-bits), and the odometer (8-bits). Output is an 8-bit vector. Answer is correct when the done bit goes high.

Assume car starts with 255 L of gas and at 0km on the odometer.

Test Case	Gas Level	Odometer	Fuel Economy	Correct?
1. Boundary Check	11111111	00000000	10101011 Remainder 1	*
2. Boundary Check	00000000	11111111	00000001	Yes
3. Random Calc.	11110000	10101010	00001011 R5	Yes

** This is the output produced by dividing by 0*

Test Case 1 produces an incorrect result. We are asking the system to divide 0/0 which is undefined. The system produces this result. The controller which uses this system must not ask for a fuel economy reading when the car has not moved.

Complete Design Test:
The entire design was tested together as part of the top-level file, the IADS. The password entered was 1111, which was accepted by the system. For *simulation purposes*, the 24-bit vector is assumed to represent the data received from the A/D converter.

This 24-bit vector is mapped to the input monitors corresponding to Door System, Gas System and Distance System, in respective Big Endian placement. The packet created by the Input Monitor includes the component address and component data. The Bus Controller begins polling the component with the lowest address and continues to sequentially poll the remaining components. The corresponding component transmitter then shifts the packet onto the bus. After the packet receiver serially loads the packet, the address and data is ready to be loaded in parallel by the address decoder. Depending on the address, the decoder transfers the data to the corresponding system module. These modules perform the data manipulation necessary to display the output correctly.

Case1: Input Vector- 111101010001111100000101

Outputs:

1. Gas System: The output is 00000010, indicating medium level which is accurate since the gas input value is 00011111 which is 31.
2. Fuel Economy: The output is 00000000 since to calculate fuel economy a change in distance and gas level is required (other than from zero).
3. Distance Projection: The output is 0000000000000000 since it needs fuel economy value.
4. Door Status: The output is 00001010 since the input is 11110101 indicating that the fourth and second doors are open. Thus, light should go open for these two. Therefore, the output contains 1's for the fourth and second door (bits 3 and 0). The most significant 4 bits are irrelevant for this output.

Case2: Input Vector- 111101110001101000001111

Case3: Input Vector- 111111110001010000101011

Case4: Input Vector- 111111100000100000111110
Case5: Input Vector- 111111100000111000111100
Case6: Input Vector- 111100000000111000111100

The outputs for these are shown in the simulation. The test cases were chosen to represent a range of values and to test the accuracy of the results both boundary as well as intermediate values were chosen.

Significant results are annotated on the waveforms attached in the Appendix.