Design Specification Document:

Abstract

This module simulates a scanner system on a gondola that will communicate with a transfer station and other scanner.

Introduction

This module simulates a scanner on a gondola that use protocol to communicate and can collects data, display currents states, send signals and receives signals between the transfer station, transfer data to the transfer station and clear the buffer of the scanner. Since we don't actually connect to the other scanner, we will assume that all the signals sent from the other scanner will goes to the transfer station and the commands sent to our scanner come from the station by switches. When the data is transferred, they will also goes to our transfer station. To make the module more realistic, the clock of the transfer station and that of the scanner are synchronized. The states of the scanner and the signal received by the station showing indicating the buffer has been used by certain amount will be displayed on the LEDR. The percentage of the buffer that has been used in 128 scale will be displayed in HEX. The data we send to the transfer station is binary numbers and they will be interpreted in a unit of 8 bits, convert to decimal and also display on the HEX.

State specification:

Current state of the scanner(Use LEDR[2:0])

LEDR[2:0] = 3'b000: When a scanner is in *LowPower* state(off state)

LEDR[2:0] = 3'b001: When a scanner is in *Standby* state(no data)

LEDR[2:0] = 3'b010: When a scanner is in *Active* state(scanning)

LEDR[2:0] = 3'b011: When a scanner is in *Idle* state(standby with data)

LEDR[2:0] = 3'b100: When a scanner is in *Transferring* state(moving data to station)

LEDR[2:0] = 3'b101: When a scanner is in *Flush* state(clear data)

Ports:

Input:

KEY[0] reset: reset the entire system to the original state, the scanner is in *LowPower* state, no data in the buffer and no data transferred to the station.

Scanner:

1. Active command: SW[0], only if active is true, the scanner can scan, send signal and transfer data, but the scanner can flush data if it's false
2. Gotostandby command: SW[3], the scanner go to *Standby* state when receiving this signal. Without this, the scanner will not be waken up
3. Startscanning command: SW[4], the scanner will start scanning only if this signal is received
4. Flush command: SW[2], the scanner will clear the buffer when this signal is received, note that this process will goes on whether or not the system is active. One bit will be delete every other clock so that clearing the buffer is twice faster than scanning
5. Transfer permit: SW[1], the scanner will transfer the data to the station if the transfer permit is received. Otherwise it will stay at *idle* state
6. Signal that the station is ready to receive data: GPIO_0[35], the scanner will send next bit only when the station is ready to receive. When it's not true, the scanner won't send data

Transfer station:

1. Active command:SW[0], only if active is true, the transfer station receive data
2. Data transferred by the scanner: GPIO_0[32], input the data sent by the scanner, one bit for each transfer clock cycle when the station is active.
3. Clock transferred by the scanner: GPIO_0[31], input the clock used by the scanner, the transfer station should receive data under same clock as the scanner

Output:

Scanner:

1. Data to be transferred to the station: GPIO_0[33], when the system is active and the station is ready to transfer data, every other clock cycle one bit will output to the transfer station, so the transferring is twice faster than scanning data.
2. Clock to be transferred to the station: GPIO_0[34], the clock used by the scanner will also output to the station so that the station receive data under the same clock
3. The number of the used bits: HEX[2:0]. the number of used bits will output to the top level module. They will be converted to the decimal value and display on the HEX using a small module in a scale of maximum 128 .
4. The current state of the scanner: LEDR[2:0]. The current state of the scanner will be output to the board and displayed on LEDR. State specification have been shown above.

5. The transfer permit has been received: LEDR[4]. The output whether the scanner have received the transfer permit will be displayed on LEDR[4]

Transfer station:

1. Receive the signal that the buffer has been used by 50%: LEDR[9]. The output whether the station have received the signal that the scanner has reached its 50% of buffer will be displayed on LEDR[9]
2. Receive the signal that the buffer has been used by 80%:LEDR[8]. The output whether the station have received the signal that the scanner has reached its 80% of buffer will be displayed on LEDR[9]
3. Receive the signal that the buffer has been used by 90%:LEDR[7]. The output whether the station have received the signal that the scanner has reached its 90% of buffer will be displayed on LEDR[9]
4. Receive the signal that the buffer has been used by 100%:LEDR[6]. The output whether the station have received the signal that the scanner has reached its 100% of buffer will be displayed on LEDR[9]
5. If the station can receive data right now:LEDR[5]. The output whether the station can receive signals will be displayed on LEDR[9]
6. The station is ready to receive next bit to the scanner: GPIO_0[30]. The ready signal will output to the scanner from GPIO_0[30] indicating that the station can receive next bit.

   Current bytes received from the scanner: HEX[5:3]. The byte that the scanner is send will output to the top level module. They will be converted to the decimal value and display on the HEX using a small module in a scale of maximum 128. Note that each byte consist of 8 bits. When the scanner is scanning, it will display the current state of the scanning, 001 for 50%, 002 for 80%, 003 for 90% and 004 for 100%. When the transfer begins, it will show 007 first indication that the scanner is sending binary numbers and then display the data that is being sent.

Major Functions

1. Scanner can collect data, send current state to the station and board, receive commands from the station and enter other states based on the commands, display current percentage used in its buffer on the board during scanning, transferring and flushing state, transfer to the station or clear data if requested.
2. Station can receive and send signals between scanner, change current state based on the signals, receive data from the scanner and display its state on the board

Functional Decomposition:

Module Scanner:

- Collect data and store them into the buffer
- Enter different states when receiving commands from the station
- In each state, the scanner will have different performance based on the state and future commands.
- In scanning state, the scanner should send different signals indicating the current state to the station
- In the transferring state, the scanner should send the data when requested from the station
- In the flushing state, the scanner should clear the buffer
- The transferring and clearing should be twice faster that collecting data

Module TransferStation:

- Send different commands based on the state of the scanner
- Indicate the scanner if the station can receive data
- Give permit to the scanner of transferring data
- Receive data from the scanner

Module HEXdisplay

- Convert 8-bit bytes and display on the HEX as decimal number

Module DE1_SoC

- Combine all the modules above so that the entire project can work properly
- Add the components of the FPGA board into the circuit so that the result of the modules can show on the board

Processing procedure:

1. Scanner in *LowPower* state after KEY[0] reset, LEDR[2:0] is off at this state.
2. Scanner receives *GoTo Standby* signal and goes to *Standby* state when SW[3] is 1, LEDR[0] is on in this state.
3. Scanner goes to *Active* state when SW[0] is 1, LEDR[1] is on in this state.
4. Scanner receives *StartScanning* signal and start scanning when SW[4] is 1, HEX[2:0] display the percentage of the buffer we are using.
5. When the scanner is 50% full, the scanner will send the signal(signal_50) to the transfer station displayed as LEDR[9], we use number 001 in HEX[5:3] to display this.
6. When the scanner is 80% full, the scanner will send the *GoToStandby(byte number 001)* signal to the other scanner and also a *PermitRequest (byte number 002)* signal to the transfer station to request transfer permit. Both signals will be displayed by LEDR[8], we use number 002 in HEX[5:3] to display this. And the transfer permit

will be given by SW[1] and will be displayed by LEDR[4].When the scanner is 90% full, the system generates *StartScanning (byte number 003)* signal to the transfer station displayed by LEDR[7], we use number 001 in HEX[5:3] to display this.

7. When the scanner is 100% full, the scanner will generates signal *(byte number 004)* to the transfer station displayed by LEDR[6], we use number 001 in HEX[5:3] to display this.
8. The scanner will go to *Idle* state after 100% full. LEDR[1:0] is on. If the permit by SW[1] is being given and the flush command by SW[2] is now, the data store in the scanner should be transferred to the station and the state changes to transferring state and LEDR[2] is on. Note that this process will goes on as long as the permit is given after the scanner is 80% full. The transfer process should be twice faster than collecting data.
9. When the data is being transferred. HEX[2:0] display the percentage of the buffer we are using. HEX[5:3] will display the number representing the type of data we are transferring. In this lab, we use byte number 007 to show that we are transferring binary numbers.
10. If the *flush* command is given by SW[2], the data from the scanner should be cleared if the scanner is full, the process is twice faster than collecting data.
11. After all the above procedures, the scanner will go back to the *LowPower* state.

Data transferred:

We transfer binary number in this lab. The data is displayed on HEX[5:3] after converting to decimal number. The number on the board should decrease by 1 each time starting from 127 until 0.