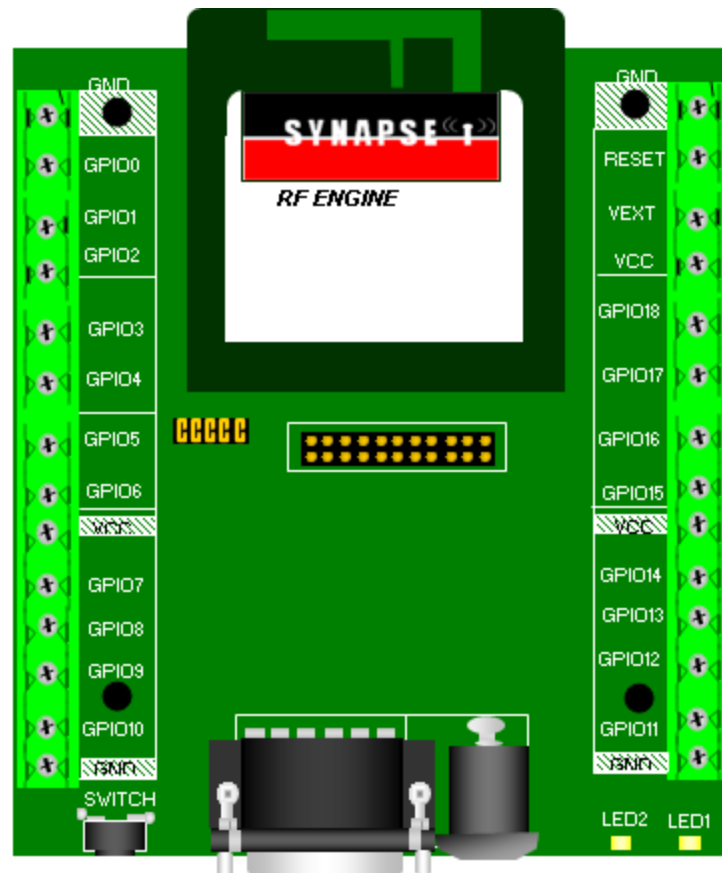


Getting Started With SNAP Technology: Wireless Embedded Lab Projects



Don Wilcher

SNAP Project: The Full Adder

A Full Adder is capable of adding 2 binary bits and displaying the resultant using discrete LEDs. The Full Adder is the basic building block for the Arithmetic Logic Unit used in Digital Calculators as well as microprocessor and microcontrollers. The following application will illustrate the functionality of the Full Adder using the RF Engine Module 2 discrete logic ICs, 2 switches and 2 LEDs. The RF Engine Module will assist in the data processing circuit application by providing the following monitoring and control functions.

- Read Status of Logic Input Switches (**A & B**)
- Direct the output pins of the RF Engine Module to control the inputs (**G1** and **G2**) of the Full Adder
- Monitor and display the status of the Logic Input Switches (**A & B**) and the Control Inputs (**G1 & G2**).

The Block Diagram for the Full Adder application is shown below.

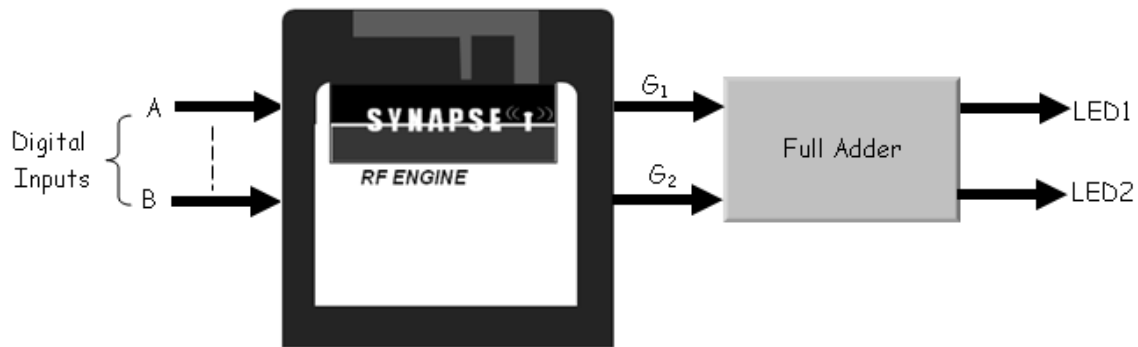
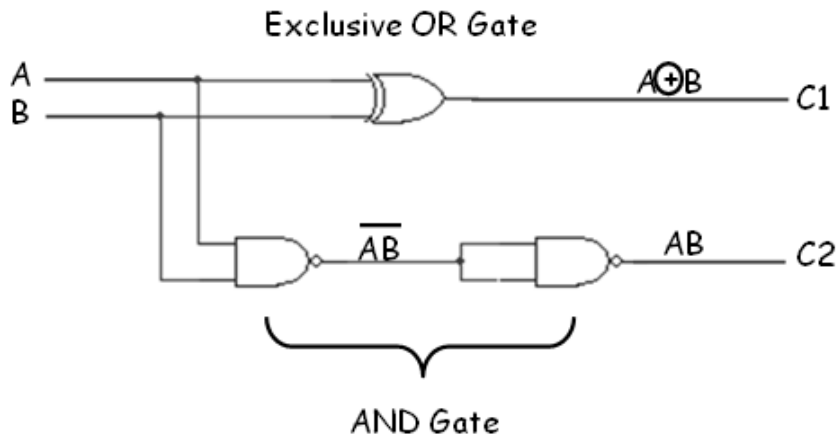


Figure 2-5. Full Adder Block Diagram

Pedagogical Concept

The full adder is capable of adding two input bits and the input carry. The half adder adds the two input bits A and B using an Exclusive-OR Gate. The input Carry (C_{in}) is added to the A and B inputs of the logic gate. To create a Full Adder, the logic inputs of A and B must be wired to an AND Gate in addition to the Exclusive -OR Gate. Figure 2-6 shows the basic Combinational Logic Circuit of the Full Adder along with its Truth Table.



A	B	C1	C2	Sum
0	0	0	0	0+0=00
0	1	0	1	0+1=01
1	0	0	1	1+0=01
1	1	1	0	1+1=10

Figure 2-6. Full Adder Schematic Circuit Diagram with Truth Table

As illustrated in the Truth Table, the “Sum” column adds the “A” and “B” inputs from the Combinational Logic Gate. Note, there is no “Carry” bit present during the first 3 addition operations. The last row in the Truth Table shows the Carry Bit of “1” as shown in the answer of “10”. In the lab exercise, the operation of the Full Adder will be validated by the use of the Truth Table shown in **Figure 2-6**.

The method in which the Full Adder was built is based on having the RF Engine Module obtain the binary data from the Active Hi Logic switches “A” and “B” and control the inputs of the Full Adder by assigning output ports of the wireless embedded device. Using the RF Engine as a wireless embedded interface enhances the Combinational Logic basic decision making function.

Part Lists

- (1) SN171 Proto Board
- (1) SN163 Bridge Demonstration Board or
- (1) SN132 SNAPstick USB Module
- (2) SPST (Single Pole Single Throw) Momentary Pushbutton Switch
- (2) 1K Ω resistor (Brown, Black, Red)
- (1) CMOS 4011 Quad NAND Gate IC
- (1) CMOS 4070 Quad Exclusive OR (XOR) Gate IC
- (2) Discrete red LEDs
- (2) 330 Ω resistors (Orange, Orange, Brown)
- (3) 1.5V Batteries
- Smart Logic Probe Kit
- Solderless Breadboard
- DMM (Digital Multimeter) (Optional)
- 22 AWG (American Wire Gauge) Solid Wire



Radio Shack sells a low cost electronics lab kit great for SNAP Wireless Embedded interfacing projects. The Catalog Number is 28-280 (Electronics Learning Lab).

Assembly and Test Procedure

1. Build the Full Adder Logic Circuit shown in **Figure 2-7** on a solderless breadboard.
2. Type the SNAPpy Script shown in **Figure 2-8** using the Portal IDE Text Editor.
3. Save the file as “Full_Adder.py”
4. Upload the script (the **image**) to the Proto Board.
5. Apply power to the Proto Board, and the Full Adder Logic Circuit.

6. Validate the Full Adder's circuit operation using the Truth Table shown in **Figure 2-6**.

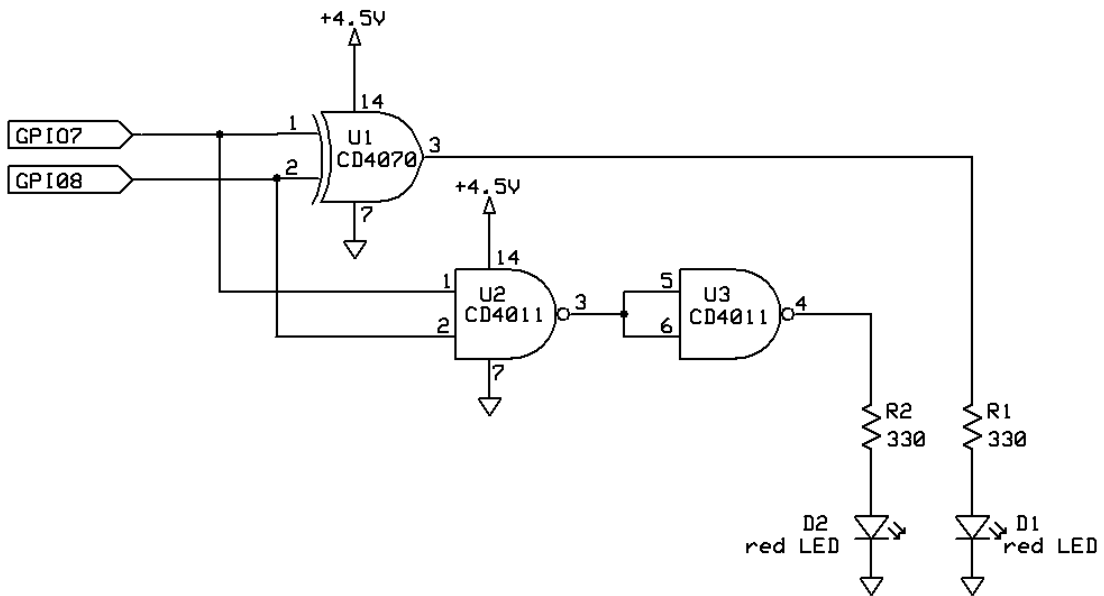
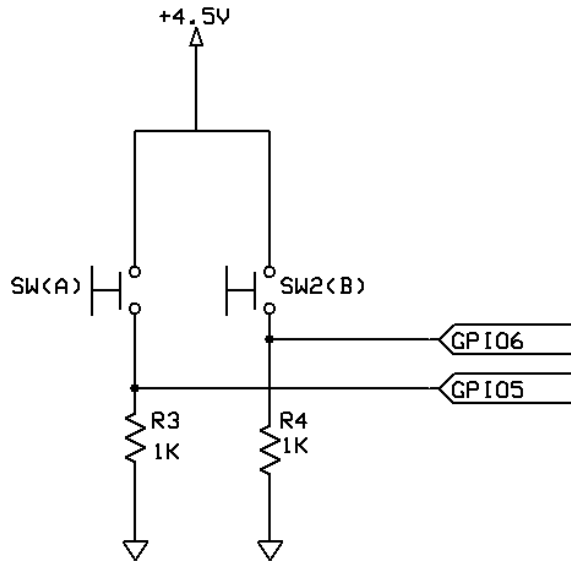


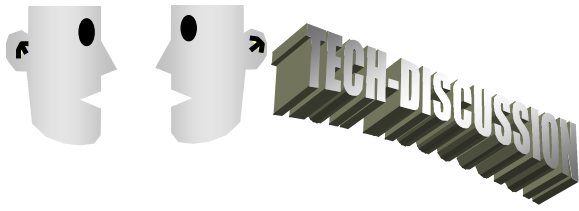
Figure 2-7. The Full Adder Circuit

```

1  """
2  This script creates a FULL ADDER with a LED Display. By pressing the appropriate logic
3  switches "A" & "B", the SUMMATION binary values will be display using discrete LEDs.
4  Also, the "status" and "event" of the Logic Switches
5  and LEDs can be read via Int STDOUT and the "print" statement.
6  """
7
8  # Use the definitions from the evalBase.py module
9  from synapse.evalBase import *
10
11 # Track the current state of the LEDs, for toggling them
12 AledState = False
13 BledState = False
14
15
16 def startupEvent():
17     """Executed when the device boots up, this initializes the hardware and begins monitoring the button."""
18     global GATE_PIN1, GATE_PIN2, B_PIN, A_PIN
19
20
21     GATE_PIN1 = 7
22     GATE_PIN2 = 8
23     B_PIN = 6
24     A_PIN = 5
25
26     # Wiring Diagram
27     #
28     #A_Sw----->|GPIO5      |
29     #B_Sw----->|GPIO6  OUTPIN |----->GPIO7(Pin 1 for 4070 & 4011 ICs)
30     #           |              |----->GPIO8(Pin 2 for 4070 & 4011 ICs)
31     #           |              |
32
33
34     writePin(GATE_PIN1, False) # Output from RFE Engine Module to Combinational Gate Input
35     setPinDir(GATE_PIN1, True)
36
37     writePin(GATE_PIN2, False) # Output from RFE Engine Module to Combinational Gate Input
38     setPinDir(GATE_PIN2, True)
39
40     setPinDir(B_PIN, False)    # Input from Active Hi Logic Sw B to RF Engine Module Input
41     monitorPin(B_PIN, True)
42
43     setPinDir(A_PIN, False)    # Input from Active Hi Logic Sw A to RF Engine Module Input
44     monitorPin(A_PIN, True)
45
46 def buttonEvent(pin, isSet):
47     """This event handler runs when the logic switch (button) is pressed"""
48     global AledState, BledState
49
50     # Take action on "press" of the button, when 'isSet' is False
51
52     # print "Pin Activated is:", pin (Statement used for PIN debug of Proto Board)
53     if pin == 5:
54         print "Button A Status is:", isSet # Check to see if Logic A Switch is pressed
55         # Logic value will be displayed in the Event Log
56         if not isSet:
57             # Latch Event of Logic Switch
58             AledState = not AledState # Toggle LED state variable
59             writePin(GATE_PIN1, AledState) # Write (Turn ON) to Output PIN with Toggled LED State
60             print "G1_INPUT is:", AledState
61
62     else:
63         # If Logic A switch is not pressed, process Logic B Switch
64         print "Button B Status is:", isSet
65         if not isSet:
66             # Toggle the LED state variable
67             BledState = not BledState
68             writePin(GATE_PIN2, BledState) # Write (Turn ON) to Output PIN with Toggled LED State
69             print "G2_INPUT is:", BledState
70
71 # Here's where we specify any "Event Handlers" we need.
72 snappyGen.setHook(SnapConstants.HOOK_STARTUP, startupEvent)
73 snappyGen.setHook(SnapConstants.HOOK_GPIN, buttonEvent)

```

Figure 2-8. The Full Adder SNAPpy Script



The Full Adder is a basic digital arithmetic circuit capable of manipulating binary digits to perform addition. The method of basic I/O interfacing using the SNAP RF Engine Module allows the SNAPpy script to capture the status and states of the Active Hi Logic Switches and the Full Adder and display their binary data within the Event Log. A 3 or 4Bit Adder can be built by duplicating the basic Combinational Logic circuit and modifying the SNAPpy Script to accommodate the additional binary data inputs.

Questions

1. Create the Truth Table for an XOR Gate
2. Write the Boolean Expression for a Full Adder Logic Circuit
3. What does CMOS stand for?
4. Draw the Logic Symbol for the Full Adder Logic Circuit
5. Re-draw the schematic diagram shown in Figure 2-7 with a 2-Input AND Gate.

For Further Investigation

Modify the Full Adder SNAPpy Script whereby the Output LEDs are not latching after each button press actuation.