myHDL Combinational Logic Elements: Demultiplexers (DEMUXs))

Steven K Armour

Table of Contents

1 Refrances

2 Libraries and Helper functions

In [1]:

```
#This notebook also uses the `(some) LaTeX environments for Jupyter`
#https://github.com/ProfFan/latex_envs wich is part of the
#jupyter_contrib_nbextensions package

from myhdl import *
from myhdlpeek import Peeker
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

from sympy import *
init_printing()

import itertools

#EE drawing tools in python from https://cdelker.bitbucket.io/SchemDraw
import SchemDraw as schem
import SchemDraw.elements as e
import SchemDraw.logic as l

#https://github.com/jrjohansson/version_information
%load_ext version_information
%version_information myhdl, myhdlpeek, numpy, pandas, matplotlib, sympy
```

Out[1]:

Software	Version
Python	3.6.2 64bit [GCC 4.4.7 20120313 (Red Hat 4.4.7-1)]
IPython	6.2.1
OS	Linux 4.15.0 30 generic x86_64 with debian stretch sid
myhdl	0.10
myhdlpeek	0.0.6
numpy	1.13.3
pandas	0.23.4
matplotlib	2.1.0
sympy	1.3
itertools	The 'itertools' distribution was not found and is required by the application
SchemDraw	0.3.0
	Sun Sep 23 18:24:14 2018 MDT

In [2]:
```
#helper  functions to read in the .v and .vhd generated files into pyth
def VerilogTextReader(loc, printresult=True):
    with open(f'{loc}.v', 'r') as vText:
        VerilogText=vText.read()
    if printresult:
        print(f'***Verilog modual from {loc}.v***\n\n', VerilogText)
    return VerilogText

def VHDLTextReader(loc, printresult=True):
    with open(f'{loc}.vhd', 'r') as vText:
        VerilogText=vText.read()
    if printresult:
        print(f'***VHDL modual from {loc}.vhd***\n\n', VerilogText)
    return VerilogText

def ConstraintXDCTextReader(loc, printresult=True):
    with open(f'{loc}.xdc', 'r') as xdcText:
        ConstraintText=xdcText.read()
    if printresult:
        print(f'***Constraint file from {loc}.xdc***\n\n', ConstraintTe
    return ConstraintText
```

In [3]:
```
def TruthTabelGenrator(BoolSymFunc):
    """
    Function to generate a truth table from a sympy boolian expression
    BoolSymFunc: sympy boolian expression
    return TT: a Truth table stored in a pandas dataframe
    """
    colsL=sorted([i for i in list(BoolSymFunc.rhs.atoms())], key=lambda
    colsR=sorted([i for i in list(BoolSymFunc.lhs.atoms())], key=lambda
    bitwidth=len(colsL)
    cols=colsL+colsR; cols

    TT=pd.DataFrame(columns=cols, index=range(2**bitwidth))

    for i in range(2**bitwidth):
        inputs=[int(j) for j in list(np.binary_repr(i, bitwidth))]
        outputs=BoolSymFunc.rhs.subs({j:v for j, v in zip(colsL, inputs
        inputs.append(int(bool(outputs)))
        TT.iloc[i]=inputs

    return TT
```

3 Demultiplexers

Definition 1 *A Demultiplexer, typically referred to as a DEMUX, is a Digital(or analog) switching unit that takes one input channel to be streamed to a single output channel from many via a control input. For single input DEMUXs with 2^n outputs, there are then n input selection signals that make up the control word to select the output channel for the input.*

Thus a DEMUX is the conjugate digital element to the MUX such that a MUX is an $N:1$ mapping device and a DEMUX is a $1:N$ mapping device.

From a behavioral standpoint DEMUXs are implemented with the same `if-elif-else` `(case)` control statements as a MUX but for each case, all outputs must be specified.

Furthermore, DEMUXs are often implemented via stacked MUXs since there governing equation is the Product SET (Cartesian product) all internal products of a MUXs SOP equation

4 1 Channel Input: 2 Channel Output demultiplexer in Gate Level Logic

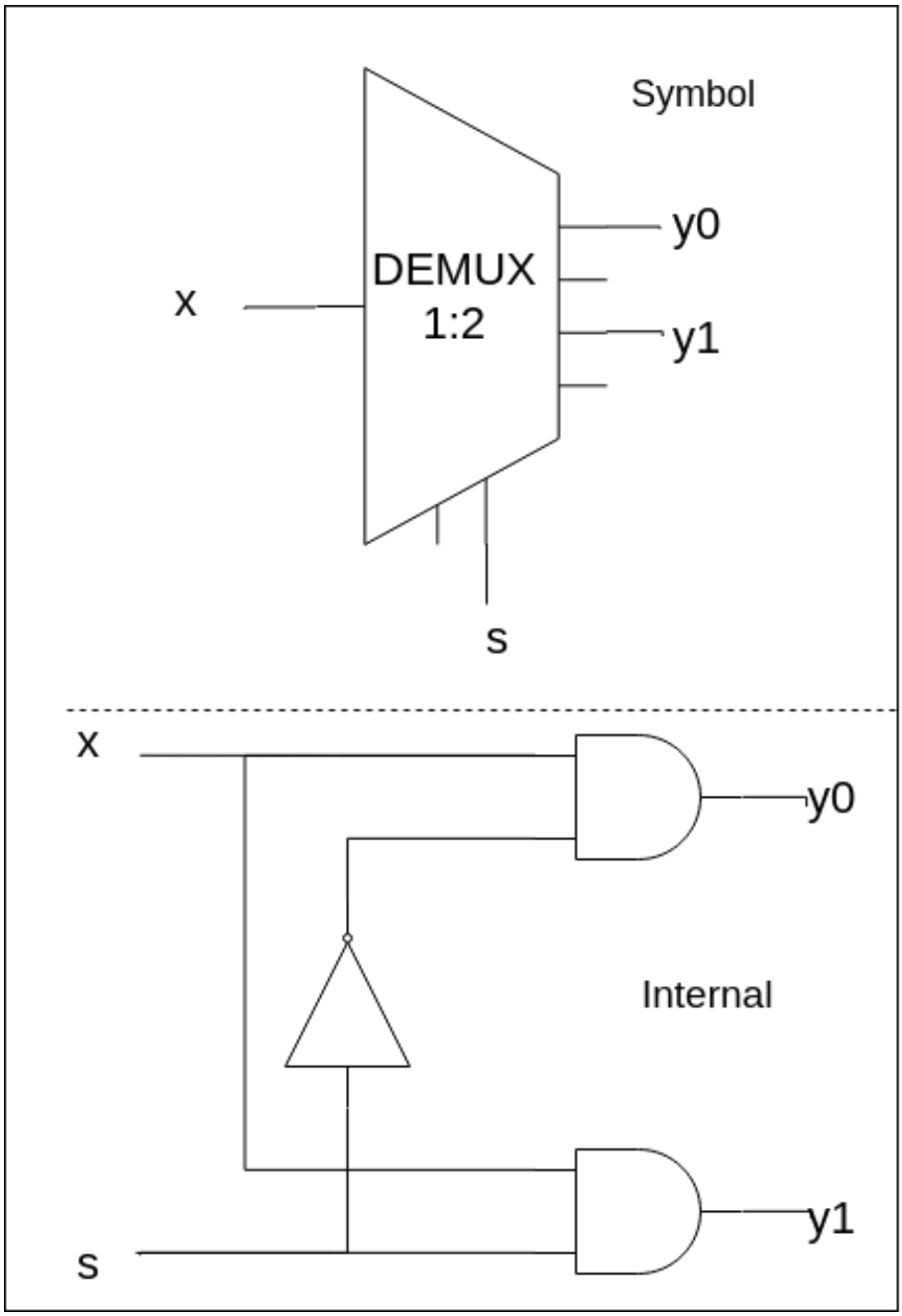


Figure 1: 1:2 DEMUX Symbol and Gate internals

4.1 Sympy Expression

In [4]:
```
x, s, y0, y1=symbols('x, s, y_0, y_1')
y12_0Eq=Eq(y0, ~s&x)
y12_1Eq=Eq(y1, s&x)
y12_0Eq, y12_1Eq
```

Out[4]: $(y_0 = x \wedge \neg s, \quad y_1 = s \wedge x)$

In [5]:
```
T0=TruthTabelGenrator(y12_0Eq)
T1=TruthTabelGenrator(y12_1Eq)
T10=pd.merge(T1, T0, how='left')
T10
```

Out[5]:

	s	x	y_1	y_0
0	0	0	0	0
1	0	1	0	1
2	1	0	0	0
3	1	1	1	0

In [6]:
```
y12_0EqN=lambdify([s, x], y12_0Eq.rhs, dummify=False)
y12_1EqN=lambdify([s, x], y12_1Eq.rhs, dummify=False)
SystmaticVals=np.array(list(itertools.product([0,1], repeat=2)))
print(SystmaticVals)
print(y12_0EqN(SystmaticVals[:, 0], SystmaticVals[:, 1]).astype(int))
print(y12_1EqN(SystmaticVals[:, 0], SystmaticVals[:, 1]).astype(int))
```

```
[[0 0]
 [0 1]
 [1 0]
 [1 1]]
[0 1 0 0]
[0 0 0 1]
```

4.2 myHDL Module

In [7]:
```
@block
def DEMUX1_2_Combo(x, s, y0, y1):
    """
    1:2 DEMUX written in full combo
    Inputs:
        x(bool): input feed
        s(bool): channel select
    Outputs:
        y0(bool): ouput channel 0
        y1(bool): ouput channel 1
    """

    @always_comb
    def logic():
        y0.next= not s and x
        y1.next= s and x

    return instances()
```

4.3 myHDL Testing

In [8]:
```
TestLen=10
SystmaticVals=list(itertools.product([0,1], repeat=2))

xTVs=np.array([i[1] for i in SystmaticVals]).astype(int)
np.random.seed(15)
xTVs=np.append(xTVs, np.random.randint(0,2, TestLen)).astype(int)

sTVs=np.array([i[0] for i in SystmaticVals]).astype(int)
#the random genrator must have a differint seed beween each generation
#call in order to produce differint values for each call
np.random.seed(16)
sTVs=np.append(sTVs, np.random.randint(0,2, TestLen)).astype(int)

TestLen=len(xTVs)
SystmaticVals, sTVs, xTVs
```

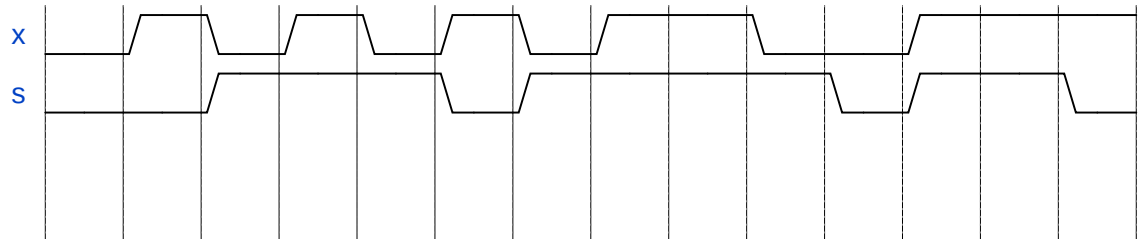
Out[8]:
```
([(0, 0), (0, 1), (1, 0), (1, 1)],
 array([0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0]),
 array([0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1]))
```

In [9]:

```
Peeker.clear()
x=Signal(bool(0)); Peeker(x, 'x')
s=Signal(bool(0)); Peeker(s, 's')
y0=Signal(bool(0)); Peeker(y0, 'y0')
y1=Signal(bool(0)); Peeker(y1, 'y1')

DUT=DEMUX1_2_Combo(x, s, y0, y1)

def DEMUX1_2_Combo_TB():
    """
    myHDL only testbench for module `DEMUX1_2_Combo`
    """

    @instance
    def stimules():
        for i in range(TestLen):
            x.next=int(xTVs[i])
            s.next=int(sTVs[i])

            yield delay(1)

        raise StopSimulation()

    return instances()

sim=Simulation(DUT, DEMUX1_2_Combo_TB(), *Peeker.instances()).run()
```

In [10]:

```
Peeker.to_wavedrom('x', 's', 'y0','y1')
```



In [11]:

```
DEMUX1_2_ComboData=Peeker.to_dataframe()
DEMUX1_2_ComboData=DEMUX1_2_ComboData[['x', 's', 'y0','y1']]
DEMUX1_2_ComboData
```

Out[11]:

	x	s	y0	y1
0	0	0	0	0
1	1	0	1	0
2	0	1	0	0
3	1	1	0	1
4	0	1	0	0
5	1	0	1	0
6	0	1	0	0
7	1	1	0	1
9	0	1	0	0
10	0	0	0	0
11	1	1	0	1
13	1	0	1	0

In [12]:

```
DEMUX1_2_ComboData['y0Ref']=DEMUX1_2_ComboData.apply(lambda row:y12_0Ec
DEMUX1_2_ComboData['y1Ref']=DEMUX1_2_ComboData.apply(lambda row:y12_1Ec
DEMUX1_2_ComboData
```

Out[12]:

	x	s	y0	y1	y0Ref	y1Ref
0	0	0	0	0	0	0
1	1	0	1	0	1	0
2	0	1	0	0	0	0
3	1	1	0	1	0	1
4	0	1	0	0	0	0
5	1	0	1	0	1	0
6	0	1	0	0	0	0
7	1	1	0	1	0	1
9	0	1	0	0	0	0
10	0	0	0	0	0	0
11	1	1	0	1	0	1
13	1	0	1	0	1	0

In [13]:
```
Test0=(DEMUX1_2_ComboData['y0']==DEMUX1_2_ComboData['y0Ref']).all()
Test1=(DEMUX1_2_ComboData['y1']==DEMUX1_2_ComboData['y1Ref']).all()
Test=Test0&Test1
print(f'Module `DEMUX1_2_Combo` works as exspected: {Test}')
```

```
Module `DEMUX1_2_Combo` works as exspected: True
```

4.4 Verilog Conversion

In [14]:
```
DUT.convert()
VerilogTextReader('DEMUX1_2_Combo');
```

```
***Verilog modual from DEMUX1_2_Combo.v***

 // File: DEMUX1_2_Combo.v
// Generated by MyHDL 0.10
// Date: Sun Sep 23 18:24:15 2018


`timescale 1ns/10ps

module DEMUX1_2_Combo (
    x,
    s,
    y0,
    y1
);
// 1:2 DEMUX written in full combo
// Inputs:
//     x(bool): input feed
//     s(bool): channel select
// Outputs:
//     y0(bool): ouput channel 0
//     y1(bool): ouput channel 1

input x;
input s;
output y0;
wire y0;
output y1;
wire y1;




assign y0 = ((!s) && x);
assign y1 = (s && x);

endmodule
```

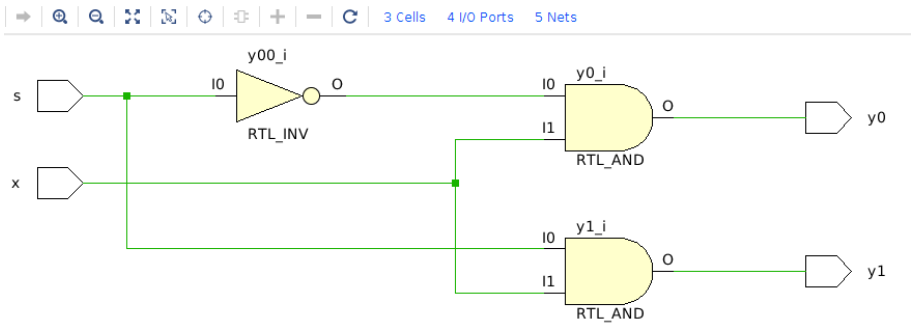


Figure 2: DEMUX1_2_Combo RTL schematic; Xilinx Vivado 2017.4

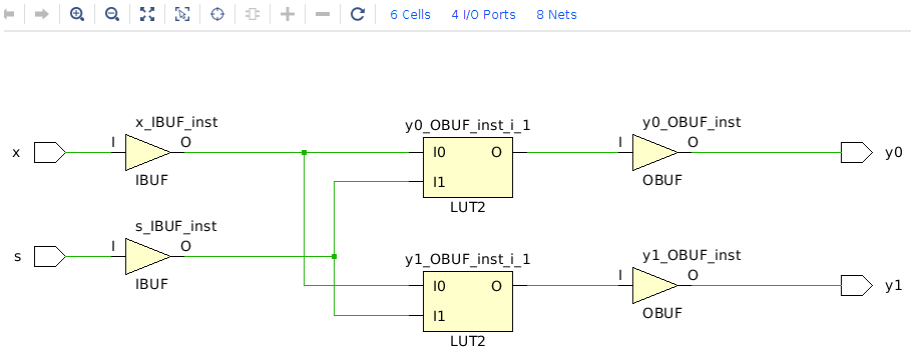


Figure 3: DEMUX1_2_Combo Synthesized Schematic; Xilinx Vivado 2017.4

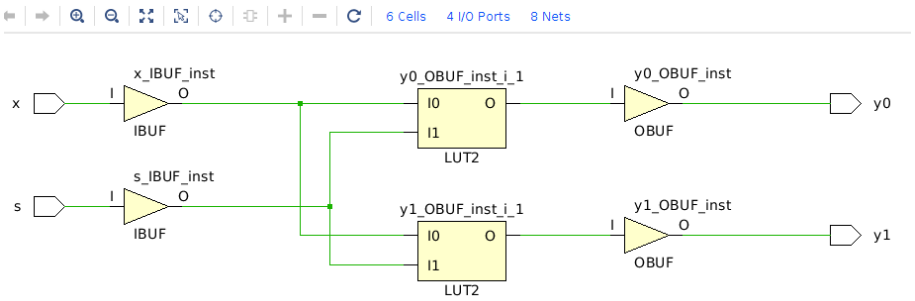


Figure 4: DEMUX1_2_Combo Implementated Schematic; Xilinx Vivado 2017.4

4.5 myHDL to Verilog Testbech

In [15]:
```
#create BitVectors
xTVs=intbv(int(''.join(xTVs.astype(str)), 2))[TestLen:]
sTVs=intbv(int(''.join(sTVs.astype(str)), 2))[TestLen:]

xTVs, bin(xTVs), sTVs, bin(sTVs)
```

Out[15]:
```
(intbv(5479), '1010101100111', intbv(3830), '111011110110')
```

In [16]:

```
@block
def DEMUX1_2_Combo_TBV():
    """
    myHDL -> testbench for module `DEMUX1_2_Combo`
    """

    x=Signal(bool(0))
    s=Signal(bool(0))
    y0=Signal(bool(0))
    y1=Signal(bool(0))

    @always_comb
    def print_data():
        print(x, s, y0, y1)

    #Test Signal Bit Vectors

    xTV=Signal(xTVs)
    sTV=Signal(sTVs)

    DUT=DEMUX1_2_Combo(x, s, y0, y1)


    @instance
    def stimules():
        for i in range(TestLen):
            x.next=int(xTV[i])
            s.next=int(sTV[i])

            yield delay(1)

        raise StopSimulation()

    return instances()

TB=DEMUX1_2_Combo_TBV()
TB.convert(hdl="Verilog", initial_values=True)
VerilogTextReader('DEMUX1_2_Combo_TBV');
```

```
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
***Verilog modual from DEMUX1_2_Combo_TBV.v***

 // File: DEMUX1_2_Combo_TBV.v
// Generated by MyHDL 0.10
// Date: Sun Sep 23 18:24:16 2018


`timescale 1ns/10ps

module DEMUX1_2_Combo_TBV (

);
// myHDL -> testbench for module `DEMUX1_2_Combo`
```

```
reg x = 0;
reg s = 0;
wire y0;
wire y1;
wire [13:0] xTV;
wire [13:0] sTV;

assign xTV = 14'd5479;
assign sTV = 14'd3830;


always @(x, y0, s, y1) begin: DEMUX1_2_COMBO_TBV_PRINT_DATA
    $write("%h", x);
    $write(" ");
    $write("%h", s);
    $write(" ");
    $write("%h", y0);
    $write(" ");
    $write("%h", y1);
    $write("\n");
end



assign y0 = ((!s) && x);
assign y1 = (s && x);


initial begin: DEMUX1_2_COMBO_TBV_STIMULES
    integer i;
    for (i=0; i<14; i=i+1) begin
        x <= xTV[i];
        s <= sTV[i];
        # 1;
    end
    $finish;
end

endmodule


/home/iridium/anaconda3/lib/python3.6/site-packages/myhdl/conversion/_t
oVerilog.py:349: ToVerilogWarning: Signal is not driven: xTV
  category=ToVerilogWarning
/home/iridium/anaconda3/lib/python3.6/site-packages/myhdl/conversion/_t
oVerilog.py:349: ToVerilogWarning: Signal is not driven: sTV
  category=ToVerilogWarning
```

4.6 PYNQ-Z1 Deployment

4.6.1 Board Circuit

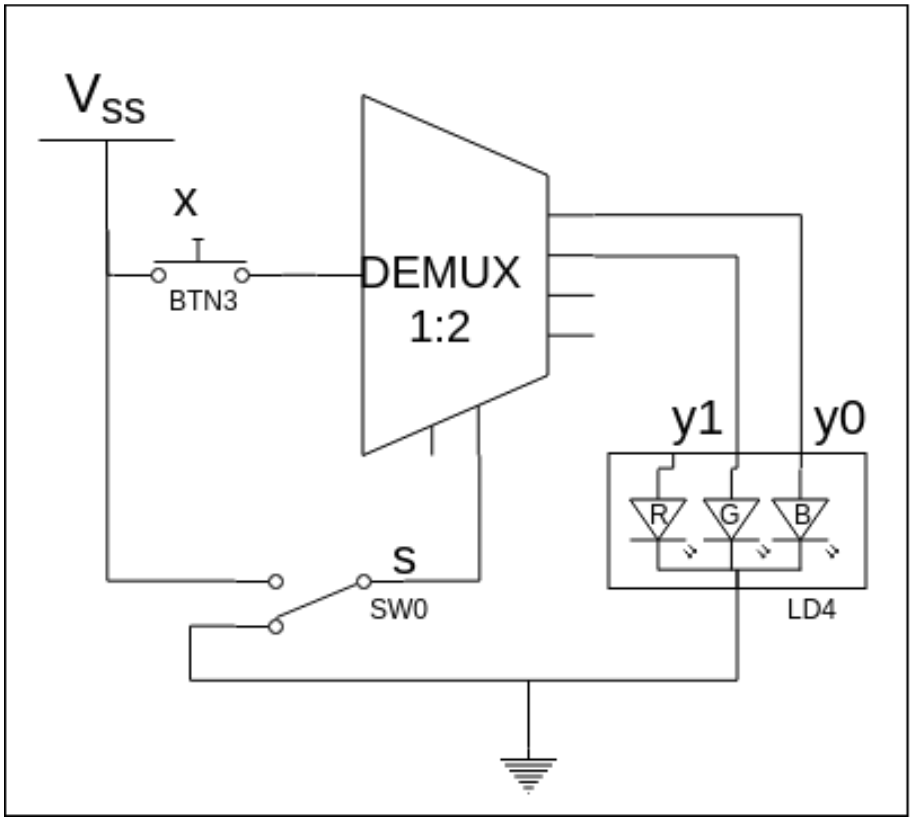


Figure 5: 1:2 DEMUX PYNQ-Z1 (Non SoC) conceptualized circuit

4.6.2 Board Constraints

In [17]:
```
ConstraintXDCTextReader('DEMUX1_2');
```

```
***Constraint file from DEMUX1_2.xdc***

 ## Switches
set_property -dict {PACKAGE_PIN M20 IOSTANDARD LVCMOS33} [get_ports
{s}]; ##SW0

## Buttons
set_property -dict {PACKAGE_PIN L19 IOSTANDARD LVCMOS33} [get_ports
{x}]; ##BT3

## RGBLEDs
set_property -dict { PACKAGE_PIN L15   IOSTANDARD LVCMOS33 } [get_ports
{y0}]; ##LD4_Blue
set_property -dict { PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports
{y1}]; ##LD4_Green
```

4.6.3 Video of Deployment

DEMUX1_2_Combo on PYNQ-Z1 (YouTube (https://www.youtube.com/watch?v=LJG4Z2kxEKE))

5 1 Channel Input:4 Channel Output demultiplexer in Gate Level Logic

5.1 Sympy Expression

In [18]:
```
x, s0, s1, y0, y1, y2, y3=symbols('x, s0, s1, y0, y1, y2, y3')
y14_0Eq=Eq(y0, ~s0&~s1&x)
y14_1Eq=Eq(y1, s0&~s1&x)
y14_2Eq=Eq(y2, ~s0&s1&x)
y14_3Eq=Eq(y3, s0&s1&x)
y14_0Eq, y14_1Eq, y14_2Eq, y14_3Eq
```

Out[18]: $(y_0 = x \wedge \neg s_0 \wedge \neg s_1,\quad y_1 = s_0 \wedge x \wedge \neg s_1,\quad y_2 = s_1 \wedge x \wedge \neg s_0,\quad y_3 = s_0 \wedge s_1 \wedge x)$

In [19]:
```
T0=TruthTabelGenrator(y14_0Eq)
T1=TruthTabelGenrator(y14_1Eq)
T2=TruthTabelGenrator(y14_2Eq)
T3=TruthTabelGenrator(y14_3Eq)
T10=pd.merge(T1, T0, how='left')
T20=pd.merge(T2, T10, how='left')
T30=pd.merge(T3, T20, how='left')
T30
```

Out[19]:

	s0	s1	x	y3	y2	y1	y0
0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1
2	0	1	0	0	0	0	0
3	0	1	1	0	1	0	0
4	1	0	0	0	0	0	0
5	1	0	1	0	0	1	0
6	1	1	0	0	0	0	0
7	1	1	1	1	0	0	0

In [20]:
```
y14_0EqN=lambdify([x, s0, s1], y14_0Eq.rhs, dummify=False)
y14_1EqN=lambdify([x, s0, s1], y14_1Eq.rhs, dummify=False)
y14_2EqN=lambdify([x, s0, s1], y14_2Eq.rhs, dummify=False)
y14_3EqN=lambdify([x, s0, s1], y14_3Eq.rhs, dummify=False)
SystmaticVals=np.array(list(itertools.product([0,1], repeat=3)))
print(SystmaticVals)
print(y14_0EqN(SystmaticVals[:, 2], SystmaticVals[:, 1], SystmaticVals[
print(y14_1EqN(SystmaticVals[:, 2], SystmaticVals[:, 1], SystmaticVals[
print(y14_2EqN(SystmaticVals[:, 2], SystmaticVals[:, 1], SystmaticVals[
print(y14_3EqN(SystmaticVals[:, 2], SystmaticVals[:, 1], SystmaticVals[
```

```
[[0 0 0]
 [0 0 1]
 [0 1 0]
 [0 1 1]
 [1 0 0]
 [1 0 1]
 [1 1 0]
 [1 1 1]]
[0 1 0 0 0 0 0 0]
[0 0 0 1 0 0 0 0]
[0 0 0 0 0 1 0 0]
[0 0 0 0 0 0 0 1]
```

5.2 myHDL Module

In [21]:
```
@block
def DEMUX1_4_Combo(x, s0, s1, y0, y1, y2, y3):
    """
    1:4 DEMUX written in full combo

    Inputs:
        x(bool): input feed
        s0(bool): channel select 0
        s1(bool): channel select 1

    Outputs:
        y0(bool): ouput channel 0
        y1(bool): ouput channel 1
        y2(bool): ouput channel 2
        y3(bool): ouput channel 3
    """

    @always_comb
    def logic():
        y0.next= (not s0) and (not s1) and x
        y1.next= s0 and (not s1) and x
        y2.next= (not s0) and s1 and x
        y3.next= s0 and s1 and x

    return instances()
```

5.3 myHDL Testing

In [22]:
```
TestLen=10
SystmaticVals=list(itertools.product([0,1], repeat=3))

xTVs=np.array([i[2] for i in SystmaticVals]).astype(int)
np.random.seed(15)
xTVs=np.append(xTVs, np.random.randint(0,2, TestLen)).astype(int)

s0TVs=np.array([i[1] for i in SystmaticVals]).astype(int)
#the random genrator must have a differint seed beween each generation
#call in order to produce differint values for each call
np.random.seed(16)
s0TVs=np.append(s0TVs, np.random.randint(0,2, TestLen)).astype(int)

s1TVs=np.array([i[0] for i in SystmaticVals]).astype(int)
#the random genrator must have a differint seed beween each generation
#call in order to produce differint values for each call
np.random.seed(17)
s1TVs=np.append(s1TVs, np.random.randint(0,2, TestLen)).astype(int)

TestLen=len(xTVs)
SystmaticVals, xTVs, s0TVs, s1TVs
```

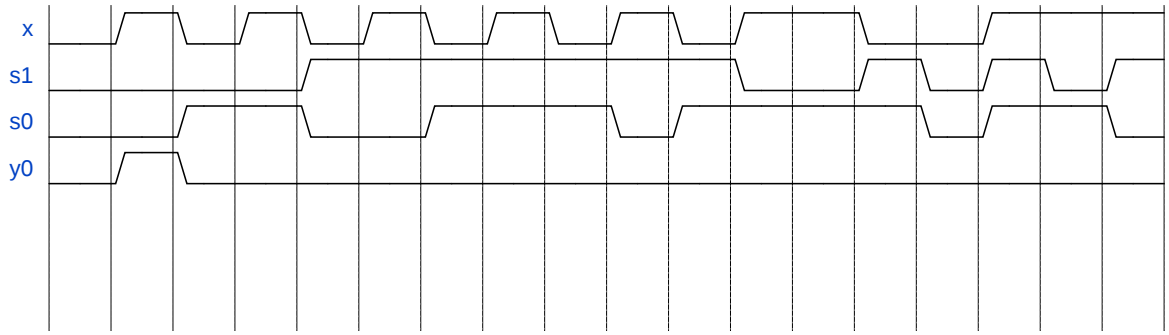
Out[22]:
```
([(0, 0, 0),
  (0, 0, 1),
  (0, 1, 0),
  (0, 1, 1),
  (1, 0, 0),
  (1, 0, 1),
  (1, 1, 0),
  (1, 1, 1)],
 array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1]),
 array([0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0]),
 array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1]))
```

In [23]:

```
Peeker.clear()
x=Signal(bool(0)); Peeker(x, 'x')
s0=Signal(bool(0)); Peeker(s0, 's0')
s1=Signal(bool(0)); Peeker(s1, 's1')
y0=Signal(bool(0)); Peeker(y0, 'y0')
y1=Signal(bool(0)); Peeker(y1, 'y1')
y2=Signal(bool(0)); Peeker(y2, 'y2')
y3=Signal(bool(0)); Peeker(y3, 'y3')


DUT=DEMUX1_4_Combo(x, s0, s1, y0, y1, y2, y3)

def DEMUX1_4_Combo_TB():
    """
    myHDL only testbench for module `DEMUX1_4_Combo`
    """

    @instance
    def stimules():
        for i in range(TestLen):
            x.next=int(xTVs[i])
            s0.next=int(s0TVs[i])
            s1.next=int(s1TVs[i])


            yield delay(1)

        raise StopSimulation()

    return instances()

sim=Simulation(DUT, DEMUX1_4_Combo_TB(), *Peeker.instances()).run()
```

In [24]:

```
Peeker.to_wavedrom('x', 's1', 's0', 'y0', 'y1', 'y2', 'y3')
```



In [25]:

```
DEMUX1_4_ComboData=Peeker.to_dataframe()
DEMUX1_4_ComboData=DEMUX1_4_ComboData[['x', 's1', 's0', 'y0', 'y1', 'y2
DEMUX1_4_ComboData
```

Out[25]:

	x	s1	s0	y0	y1	y2	y3
0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0
2	0	0	1	0	0	0	0
3	1	0	1	0	1	0	0
4	0	1	0	0	0	0	0
5	1	1	0	0	0	1	0
6	0	1	1	0	0	0	0
7	1	1	1	0	0	0	1
8	0	1	1	0	0	0	0
9	1	1	0	0	0	1	0
10	0	1	1	0	0	0	0
11	1	0	1	0	1	0	0
13	0	1	1	0	0	0	0
14	0	0	0	0	0	0	0
15	1	1	1	0	0	0	1
16	1	0	1	0	1	0	0
17	1	1	0	0	0	1	0

In [26]:
```
DEMUX1_4_ComboData['y0Ref']=DEMUX1_4_ComboData.apply(lambda row:y14_0Ec
DEMUX1_4_ComboData['y1Ref']=DEMUX1_4_ComboData.apply(lambda row:y14_1Ec
DEMUX1_4_ComboData['y2Ref']=DEMUX1_4_ComboData.apply(lambda row:y14_2Ec
DEMUX1_4_ComboData['y3Ref']=DEMUX1_4_ComboData.apply(lambda row:y14_3Ec
DEMUX1_4_ComboData
```

Out[26]:

	x	s1	s0	y0	y1	y2	y3	y0Ref	y1Ref	y2Ref	y3Ref
0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	1	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0
3	1	0	1	0	1	0	0	0	1	0	0
4	0	1	0	0	0	0	0	0	0	0	0
5	1	1	0	0	0	1	0	0	0	1	0
6	0	1	1	0	0	0	0	0	0	0	0
7	1	1	1	0	0	0	1	0	0	0	1
8	0	1	1	0	0	0	0	0	0	0	0
9	1	1	0	0	0	1	0	0	0	1	0
10	0	1	1	0	0	0	0	0	0	0	0
11	1	0	1	0	1	0	0	0	1	0	0
13	0	1	1	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0
15	1	1	1	0	0	0	1	0	0	0	1
16	1	0	1	0	1	0	0	0	1	0	0
17	1	1	0	0	0	1	0	0	0	1	0

In [27]:
```
Test0=(DEMUX1_4_ComboData['y0']==DEMUX1_4_ComboData['y0Ref']).all()
Test1=(DEMUX1_4_ComboData['y1']==DEMUX1_4_ComboData['y1Ref']).all()
Test2=(DEMUX1_4_ComboData['y2']==DEMUX1_4_ComboData['y2Ref']).all()
Test3=(DEMUX1_4_ComboData['y3']==DEMUX1_4_ComboData['y3Ref']).all()
Test=Test0&Test1&Test2&Test3
print(f'Module `DEMUX1_4_Combo` works as exspected: {Test}')
```

```
Module `DEMUX1_4_Combo` works as exspected: True
```

5.4 Verilog Conversion

In [28]:
```
DUT.convert()
VerilogTextReader('DEMUX1_4_Combo');
```

```
***Verilog modual from DEMUX1_4_Combo.v***

 // File: DEMUX1_4_Combo.v
// Generated by MyHDL 0.10
// Date: Sun Sep 23 18:24:17 2018


`timescale 1ns/10ps

module DEMUX1_4_Combo (
    x,
    s0,
    s1,
    y0,
    y1,
    y2,
    y3
);
// 1:4 DEMUX written in full combo
//
// Inputs:
//     x(bool): input feed
//     s0(bool): channel select 0
//     s1(bool): channel select 1
//
// Outputs:
//     y0(bool): ouput channel 0
//     y1(bool): ouput channel 1
//     y2(bool): ouput channel 2
//     y3(bool): ouput channel 3

input x;
input s0;
input s1;
output y0;
wire y0;
output y1;
wire y1;
output y2;
wire y2;
output y3;
wire y3;




assign y0 = ((!s0) && (!s1) && x);
assign y1 = (s0 && (!s1) && x);
assign y2 = ((!s0) && s1 && x);
assign y3 = (s0 && s1 && x);

endmodule
```

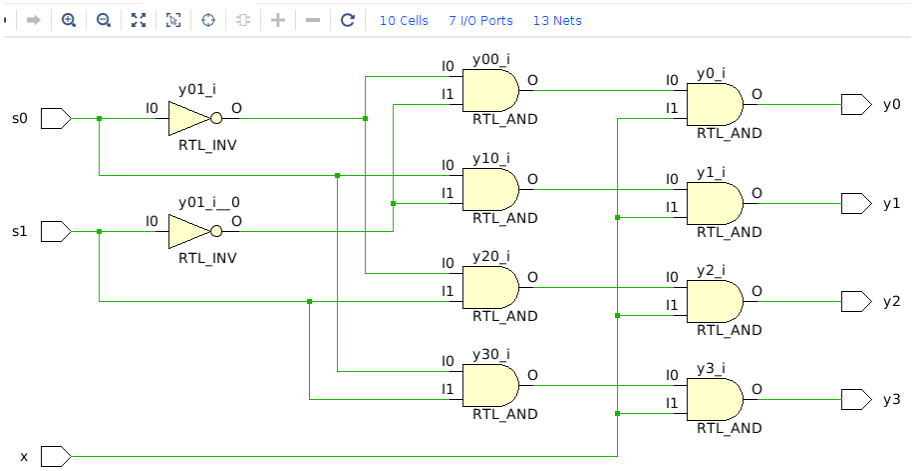


Figure 6: DEMUX1_4_Combo RTL schematic; Xilinx Vivado 2017.4

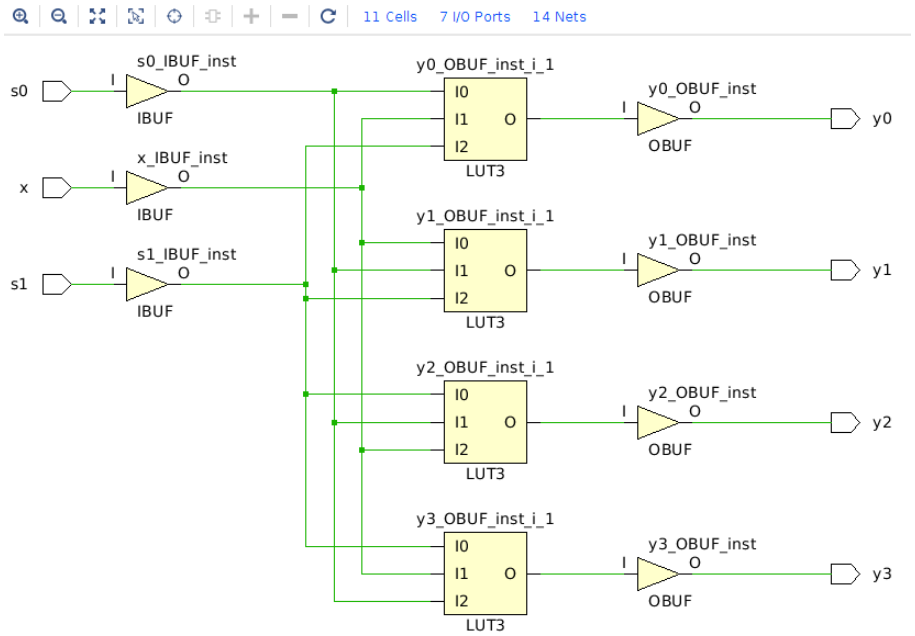


Figure 7: DEMUX1_4_Combo Synthesized Schematic; Xilinx Vivado 2017.4

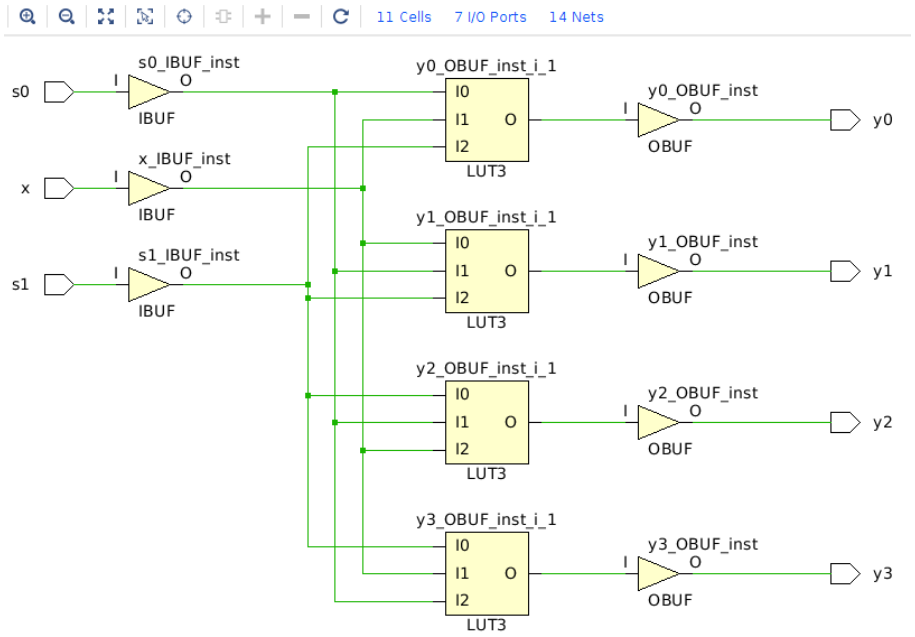


Figure 8: DEMUX1_4_Combo Implementated Schematic; Xilinx Vivado 2017.4

5.5 myHDL to Verilog Testbench

In [29]:
```
#create BitVectors
xTVs=intbv(int(''.join(xTVs.astype(str)), 2))[TestLen:]
s0TVs=intbv(int(''.join(s0TVs.astype(str)), 2))[TestLen:]
s1TVs=intbv(int(''.join(s1TVs.astype(str)), 2))[TestLen:]


xTVs, bin(xTVs), s0TVs, bin(s0TVs), s1TVs, bin(s1TVs)
```

Out[29]:
```
(intbv(87399),
 '10101010101100111',
 intbv(52982),
 '1100111011110110',
 intbv(16277),
 '11111110010101')
```

In [30]:

```
@block
def DEMUX1_4_Combo_TBV():
    """
    myHDL -> testbench for module `DEMUX1_4_Combo`
    """

    x=Signal(bool(0))
    s0=Signal(bool(0))
    s1=Signal(bool(0))
    y0=Signal(bool(0))
    y1=Signal(bool(0))
    y2=Signal(bool(0))
    y3=Signal(bool(0))

    @always_comb
    def print_data():
        print(x, s0, s1, y0, y1, y2, y3)

    #Test Signal Bit Vectors

    xTV=Signal(xTVs)
    s0TV=Signal(s0TVs)
    s1TV=Signal(s1TVs)

    DUT=DEMUX1_4_Combo(x, s0, s1, y0, y1, y2, y3)


    @instance
    def stimules():
        for i in range(TestLen):
            x.next=int(xTV[i])
            s0.next=int(s0TV[i])
            s1.next=int(s1TV[i])

            yield delay(1)

        raise StopSimulation()

    return instances()

TB=DEMUX1_4_Combo_TBV()
TB.convert(hdl="Verilog", initial_values=True)
VerilogTextReader('DEMUX1_4_Combo_TBV');
```

```
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
***Verilog modual from DEMUX1_4_Combo_TBV.v***

 // File: DEMUX1_4_Combo_TBV.v
// Generated by MyHDL 0.10
// Date: Sun Sep 23 18:24:18 2018
```

```
`timescale 1ns/10ps

module DEMUX1_4_Combo_TBV (

);
// myHDL -> testbench for module `DEMUX1_4_Combo`


reg x = 0;
wire y0;
wire y1;
reg s0 = 0;
reg s1 = 0;
wire y2;
wire y3;
wire [17:0] xTV;
wire [17:0] s0TV;
wire [17:0] s1TV;

assign xTV = 18'd87399;
assign s0TV = 18'd52982;
assign s1TV = 18'd16277;


always @(x, y0, s1, s0, y3, y2, y1) begin: DEMUX1_4_COMBO_TBV_PRINT_DAT
A
    $write("%h", x);
    $write(" ");
    $write("%h", s0);
    $write(" ");
    $write("%h", s1);
    $write(" ");
    $write("%h", y0);
    $write(" ");
    $write("%h", y1);
    $write(" ");
    $write("%h", y2);
    $write(" ");
    $write("%h", y3);
    $write("\n");
end



assign y0 = ((!s0) && (!s1) && x);
assign y1 = (s0 && (!s1) && x);
assign y2 = ((!s0) && s1 && x);
assign y3 = (s0 && s1 && x);


initial begin: DEMUX1_4_COMBO_TBV_STIMULES
    integer i;
    for (i=0; i<18; i=i+1) begin
        x <= xTV[i];
        s0 <= s0TV[i];
```

```
        s1 <= s1TV[i];
        # 1;
    end
    $finish;
end

endmodule


/home/iridium/anaconda3/lib/python3.6/site-packages/myhdl/conversion/_t
oVerilog.py:349: ToVerilogWarning: Signal is not driven: xTV
  category=ToVerilogWarning
/home/iridium/anaconda3/lib/python3.6/site-packages/myhdl/conversion/_t
oVerilog.py:349: ToVerilogWarning: Signal is not driven: s0TV
  category=ToVerilogWarning
/home/iridium/anaconda3/lib/python3.6/site-packages/myhdl/conversion/_t
oVerilog.py:349: ToVerilogWarning: Signal is not driven: s1TV
  category=ToVerilogWarning
```

5.6 PYNQ-Z1 Deployment

5.6.1 Board Circuit

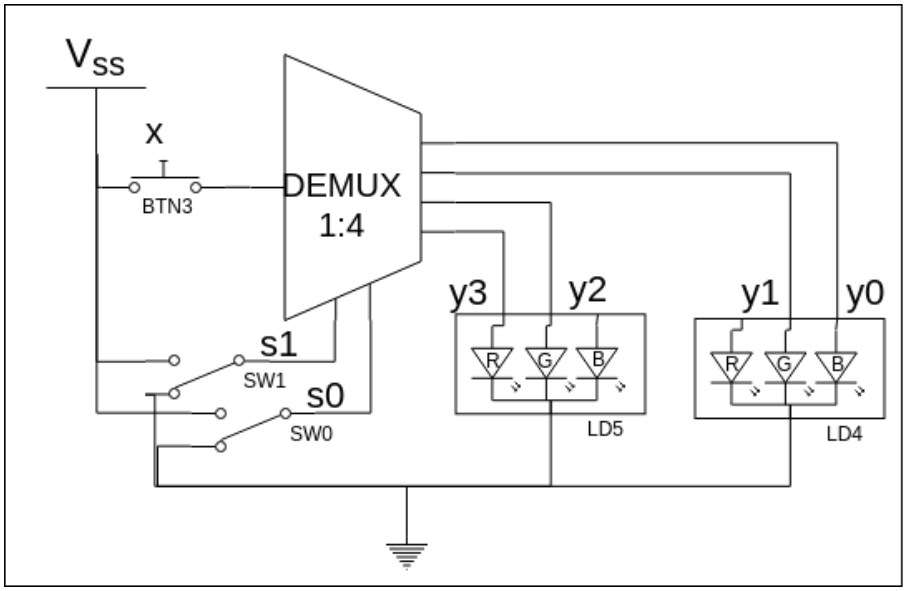


Figure 9: 1:4 DEMUX PYNQ-Z1 (Non SoC) conceptualized circuit

5.6.2 Board Constraints

In [31]:
```
ConstraintXDCTextReader('DEMUX1_4');
```

```
***Constraint file from DEMUX1_4.xdc***

 ## Switches
set_property -dict {PACKAGE_PIN M20 IOSTANDARD LVCMOS33} [get_ports {s
0}]; ##SW0
set_property -dict {PACKAGE_PIN M19 IOSTANDARD LVCMOS33} [get_ports {s
1}]; ##SW1

## Buttons

set_property -dict {PACKAGE_PIN L19 IOSTANDARD LVCMOS33} [get_ports
{x}]; ##BT3


## RGBLEDs
set_property -dict { PACKAGE_PIN L15   IOSTANDARD LVCMOS33 } [get_ports
{y0}]; ##LD4_Blue
set_property -dict { PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports
{y1}]; ##LD4_Green
set_property -dict { PACKAGE_PIN L14   IOSTANDARD LVCMOS33 } [get_ports
{y2}]; ##LD5_Green
set_property -dict { PACKAGE_PIN M15   IOSTANDARD LVCMOS33 } [get_ports
{y3}]; ##LD5_Red
```

5.6.3 Video of Deployment

DEMUX1_4_Combo on PYNQ-Z1 (YouTube (https://www.youtube.com/watch?v=VT5Iyy8dMdg))

6 1 Channel Input:4 Channel Output demultiplexer via DEMUX Stacking

6.1 myHDL Module

In [32]:

```
@block
def DEMUX1_4_DMS(x, s0, s1, y0, y1, y2, y3):
    """
    1:4 DEMUX via DEMUX Stacking
    Inputs:
        x(bool): input feed
        s0(bool): channel select 0
        s1(bool): channel select 1

    Outputs:
        y0(bool): ouput channel 0
        y1(bool): ouput channel 1
        y2(bool): ouput channel 2
        y3(bool): ouput channel 3
    """

    s0_y0y1_WIRE=Signal(bool(0))
    s0_y2y3_WIRE=Signal(bool(0))
    x_s1_DEMUX=DEMUX1_2_Combo(x, s1, s0_y0y1_WIRE, s0_y2y3_WIRE)

    s1_y0y1_DEMUX=DEMUX1_2_Combo(s0_y0y1_WIRE, s0, y0, y1)
    s1_y2y3_DEMUX=DEMUX1_2_Combo(s0_y2y3_WIRE, s0, y2, y3)

    return instances()
```

6.2 myHDL Testing

In [33]:

```
TestLen=10
SystmaticVals=list(itertools.product([0,1], repeat=3))

xTVs=np.array([i[2] for i in SystmaticVals]).astype(int)
np.random.seed(15)
xTVs=np.append(xTVs, np.random.randint(0,2, TestLen)).astype(int)

s0TVs=np.array([i[1] for i in SystmaticVals]).astype(int)
#the random genrator must have a differint seed beween each generation
#call in order to produce differint values for each call
np.random.seed(16)
s0TVs=np.append(s0TVs, np.random.randint(0,2, TestLen)).astype(int)

s1TVs=np.array([i[0] for i in SystmaticVals]).astype(int)
#the random genrator must have a differint seed beween each generation
#call in order to produce differint values for each call
np.random.seed(17)
s1TVs=np.append(s1TVs, np.random.randint(0,2, TestLen)).astype(int)

TestLen=len(xTVs)
SystmaticVals, xTVs, s0TVs, s1TVs
```

Out[33]:

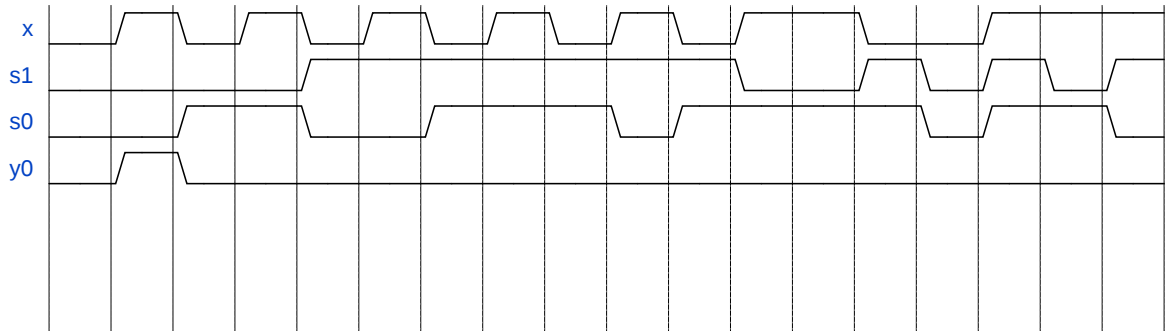
```
([(0, 0, 0),
  (0, 0, 1),
  (0, 1, 0),
  (0, 1, 1),
  (1, 0, 0),
  (1, 0, 1),
  (1, 1, 0),
  (1, 1, 1)],
 array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1]),
 array([0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0]),
 array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1]))
```

In [34]:

```
Peeker.clear()
x=Signal(bool(0)); Peeker(x, 'x')
s0=Signal(bool(0)); Peeker(s0, 's0')
s1=Signal(bool(0)); Peeker(s1, 's1')
y0=Signal(bool(0)); Peeker(y0, 'y0')
y1=Signal(bool(0)); Peeker(y1, 'y1')
y2=Signal(bool(0)); Peeker(y2, 'y2')
y3=Signal(bool(0)); Peeker(y3, 'y3')


DUT=DEMUX1_4_DMS(x, s0, s1, y0, y1, y2, y3)

def DEMUX1_4_DMS_TB():
    """
    myHDL only testbench for module `DEMUX1_4_DMS`
    """

    @instance
    def stimules():
        for i in range(TestLen):
            x.next=int(xTVs[i])
            s0.next=int(s0TVs[i])
            s1.next=int(s1TVs[i])


            yield delay(1)

        raise StopSimulation()

    return instances()

sim=Simulation(DUT, DEMUX1_4_DMS_TB(), *Peeker.instances()).run()
```

In [35]:

```
Peeker.to_wavedrom('x', 's1', 's0', 'y0', 'y1', 'y2', 'y3')
```



In [36]:
```
DEMUX1_4_DMSData=Peeker.to_dataframe()
DEMUX1_4_DMSData=DEMUX1_4_DMSData[['x', 's1', 's0', 'y0', 'y1', 'y2', '
DEMUX1_4_DMSData
```

Out[36]:

	x	s1	s0	y0	y1	y2	y3
0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0
2	0	0	1	0	0	0	0
3	1	0	1	0	1	0	0
4	0	1	0	0	0	0	0
5	1	1	0	0	0	1	0
6	0	1	1	0	0	0	0
7	1	1	1	0	0	0	1
8	0	1	1	0	0	0	0
9	1	1	0	0	0	1	0
10	0	1	1	0	0	0	0
11	1	0	1	0	1	0	0
13	0	1	1	0	0	0	0
14	0	0	0	0	0	0	0
15	1	1	1	0	0	0	1
16	1	0	1	0	1	0	0
17	1	1	0	0	0	1	0

In [37]:
```
Test=DEMUX1_4_DMSData==DEMUX1_4_ComboData[['x', 's1', 's0', 'y0', 'y1',
Test=Test.all().all()
print(f'DEMUX1_4_DMS equivlinet to DEMUX1_4_Combo: {Test}')
```

```
DEMUX1_4_DMS equivlinet to DEMUX1_4_Combo: True
```

6.3 Verilog Conversion

In [38]:
```
DUT.convert()
VerilogTextReader('DEMUX1_4_DMS');
```

Verilog modual from DEMUX1_4_DMS.v

```
 // File: DEMUX1_4_DMS.v
// Generated by MyHDL 0.10
// Date: Sun Sep 23 18:24:20 2018


`timescale 1ns/10ps

module DEMUX1_4_DMS (
    x,
    s0,
    s1,
    y0,
    y1,
    y2,
    y3
);
// 1:4 DEMUX via DEMUX Stacking
// Inputs:
//     x(bool): input feed
//     s0(bool): channel select 0
//     s1(bool): channel select 1
//
// Outputs:
//     y0(bool): ouput channel 0
//     y1(bool): ouput channel 1
//     y2(bool): ouput channel 2
//     y3(bool): ouput channel 3

input x;
input s0;
input s1;
output y0;
wire y0;
output y1;
wire y1;
output y2;
wire y2;
output y3;
wire y3;

wire s0_y2y3_WIRE;
wire s0_y0y1_WIRE;



assign s0_y0y1_WIRE = ((!s1) && x);
assign s0_y2y3_WIRE = (s1 && x);



assign y0 = ((!s0) && s0_y0y1_WIRE);
```

```
assign y1 = (s0 && s0_y0y1_WIRE);


assign y2 = ((!s0) && s0_y2y3_WIRE);
assign y3 = (s0 && s0_y2y3_WIRE);

endmodule
```

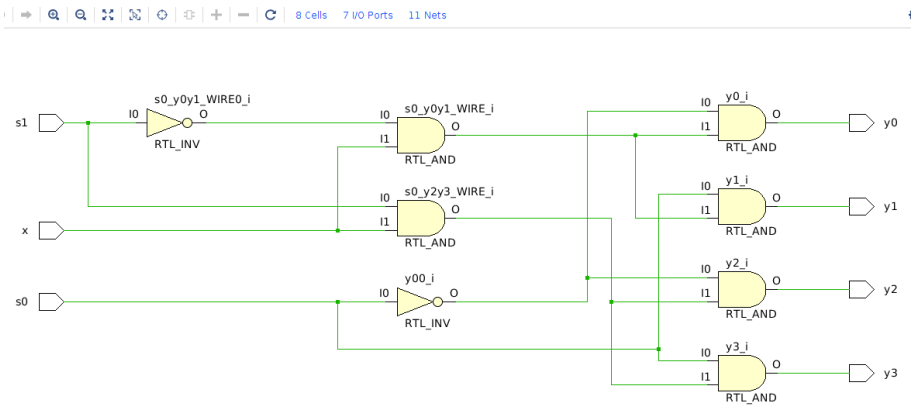


Figure 10: DEMUX1_4_DMS RTL schematic; Xilinx Vivado 2017.4

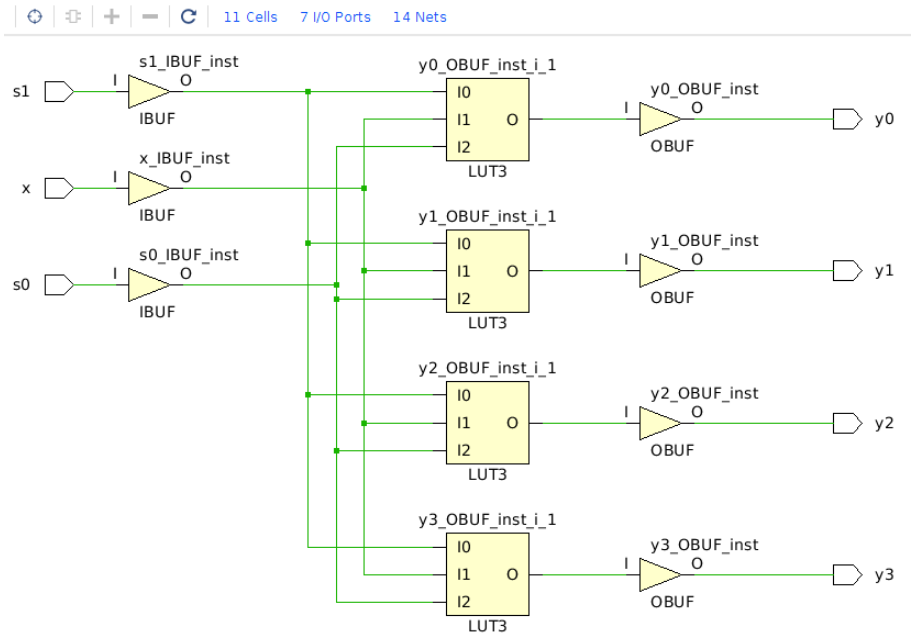


Figure 11: DEMUX1_4_DMS Synthesized Schematic; Xilinx Vivado 2017.4

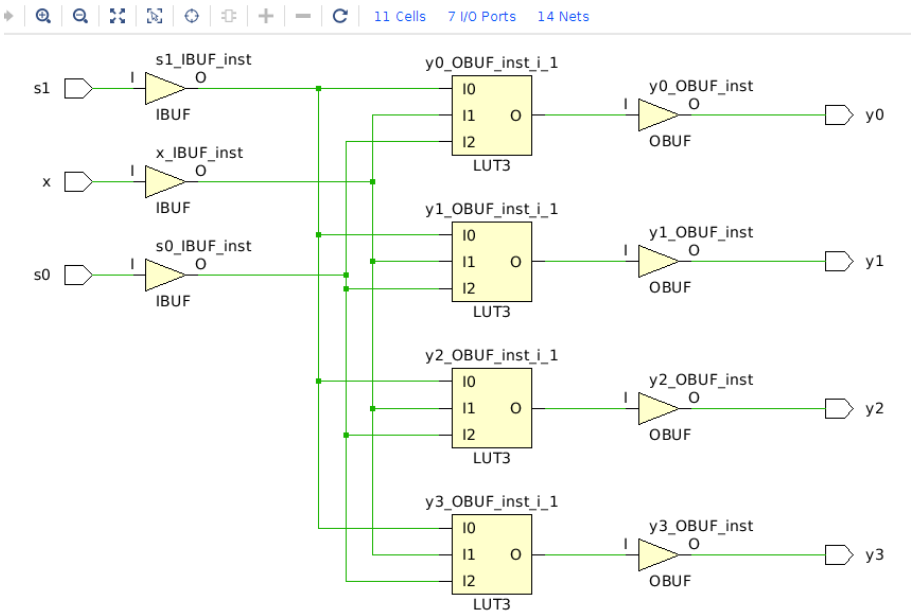


Figure 12: DEMUX1_4_DMS Implementated Schematic; Xilinx Vivado 2017.4

6.4 myHDL to Verilog Testbench

In [39]:
```
#create BitVectors
xTVs=intbv(int(''.join(xTVs.astype(str)), 2))[TestLen:]
s0TVs=intbv(int(''.join(s0TVs.astype(str)), 2))[TestLen:]
s1TVs=intbv(int(''.join(s1TVs.astype(str)), 2))[TestLen:]


xTVs, bin(xTVs), s0TVs, bin(s0TVs), s1TVs, bin(s1TVs)
```

Out[39]:
```
(intbv(87399),
 '10101010101100111',
 intbv(52982),
 '1100111011110110',
 intbv(16277),
 '11111110010101')
```

In [40]:

```
@block
def DEMUX1_4_DMS_TBV():
    """
    myHDL -> testbench for module `DEMUX1_4_DMS`
    """

    x=Signal(bool(0))
    s0=Signal(bool(0))
    s1=Signal(bool(0))
    y0=Signal(bool(0))
    y1=Signal(bool(0))
    y2=Signal(bool(0))
    y3=Signal(bool(0))

    @always_comb
    def print_data():
        print(x, s0, s1, y0, y1, y2, y3)

    #Test Signal Bit Vectors

    xTV=Signal(xTVs)
    s0TV=Signal(s0TVs)
    s1TV=Signal(s1TVs)

    DUT=DEMUX1_4_DMS(x, s0, s1, y0, y1, y2, y3)


    @instance
    def stimules():
        for i in range(TestLen):
            x.next=int(xTV[i])
            s0.next=int(s0TV[i])
            s1.next=int(s1TV[i])

            yield delay(1)

        raise StopSimulation()

    return instances()

TB=DEMUX1_4_DMS_TBV()
TB.convert(hdl="Verilog", initial_values=True)
VerilogTextReader('DEMUX1_4_DMS_TBV');
```

```
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
***Verilog modual from DEMUX1_4_DMS_TBV.v***

 // File: DEMUX1_4_DMS_TBV.v
// Generated by MyHDL 0.10
// Date: Sun Sep 23 18:24:21 2018
```

```
`timescale 1ns/10ps

module DEMUX1_4_DMS_TBV (

);
```

6.5 PYNQ-Z1 Deployment

6.5.1 Board Circuit

See Board Circuit for "1 Channel Input:4 Channel Output demultiplexer in Gate Level Logic"

6.5.2 Board Constraint

uses same 'DEMUX1_4.xdc' as "# 1 Channel Input:4 Channel Output demultiplexer in Gate Level Logic"

6.5.3 Video of Deployment

DEMUX1_4_DMS on PYNQ-Z1 (YouTube (https://www.youtube.com/watch?v=8A9iDf4nh74))

7 1:2 DEMUX via Behavioral IF

7.1 myHDL Module

In [41]:
```
@block
def DEMUX1_2_B(x, s, y0, y1):
    """
    1:2 DMUX in behavioral
    Inputs:
        x(bool): input feed
        s(bool): channel select
    Outputs:
        y0(bool): ouput channel 0
        y1(bool): ouput channel 1
    """

    @always_comb
    def logic():
        if s==0:
            #take note that since we have
            #two ouputs there next state values
            #must both be set, else the last
            #value will presist till it changes
            y0.next=x
            y1.next=0
        else:
            y0.next=0
            y1.next=x

    return instances()
```

7.2 myHDL Testing

In [42]:
```
TestLen=10
SystmaticVals=list(itertools.product([0,1], repeat=2))

xTVs=np.array([i[1] for i in SystmaticVals]).astype(int)
np.random.seed(15)
xTVs=np.append(xTVs, np.random.randint(0,2, TestLen)).astype(int)

sTVs=np.array([i[0] for i in SystmaticVals]).astype(int)
#the random genrator must have a differint seed beween each generation
#call in order to produce differint values for each call
np.random.seed(16)
sTVs=np.append(sTVs, np.random.randint(0,2, TestLen)).astype(int)

TestLen=len(xTVs)
SystmaticVals, sTVs, xTVs
```

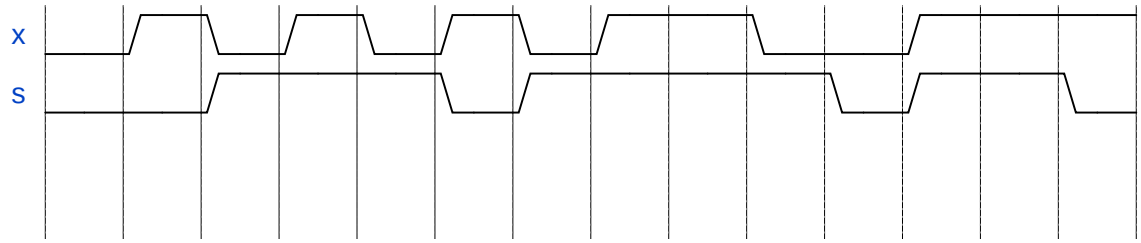
Out[42]:
```
([(0, 0), (0, 1), (1, 0), (1, 1)],
 array([0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0]),
 array([0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1]))
```

In [43]:

```
Peeker.clear()
x=Signal(bool(0)); Peeker(x, 'x')
s=Signal(bool(0)); Peeker(s, 's')
y0=Signal(bool(0)); Peeker(y0, 'y0')
y1=Signal(bool(0)); Peeker(y1, 'y1')

DUT=DEMUX1_2_B(x, s, y0, y1)

def DEMUX1_2_B_TB():
    """
    myHDL only testbench for module `DEMUX1_2_B`
    """

    @instance
    def stimules():
        for i in range(TestLen):
            x.next=int(xTVs[i])
            s.next=int(sTVs[i])

            yield delay(1)

        raise StopSimulation()

    return instances()

sim=Simulation(DUT, DEMUX1_2_B_TB(), *Peeker.instances()).run()
```

In [44]:

```
Peeker.to_wavedrom('x', 's', 'y0','y1')
```



In [45]:
```
DEMUX1_2_BData=Peeker.to_dataframe()
DEMUX1_2_BData=DEMUX1_2_BData[['x', 's', 'y0','y1']]
DEMUX1_2_BData
```

Out[45]:

	x	s	y0	y1
0	0	0	0	0
1	1	0	1	0
2	0	1	0	0
3	1	1	0	1
4	0	1	0	0
5	1	0	1	0
6	0	1	0	0
7	1	1	0	1
9	0	1	0	0
10	0	0	0	0
11	1	1	0	1
13	1	0	1	0

In [46]:
```
Test=DEMUX1_2_BData==DEMUX1_2_ComboData[['x', 's', 'y0','y1']]
Test=Test.all().all()
print(f'DEMUX1_2_BD is equivlent to DEMUX1_2_Combo: {Test}')
```

```
DEMUX1_2_BD is equivlent to DEMUX1_2_Combo: True
```

7.3 Verilog Conversion

In [47]:

```
DUT.convert()
VerilogTextReader('DEMUX1_2_B');
```

Verilog modual from DEMUX1_2_B.v

```
 // File: DEMUX1_2_B.v
// Generated by MyHDL 0.10
// Date: Sun Sep 23 18:24:23 2018


`timescale 1ns/10ps

module DEMUX1_2_B (
    x,
    s,
    y0,
    y1
);
// 1:2 DMUX in behavioral
// Inputs:
//     x(bool): input feed
//     s(bool): channel select
// Outputs:
//     y0(bool): ouput channel 0
//     y1(bool): ouput channel 1

input x;
input s;
output y0;
reg y0;
output y1;
reg y1;




always @(s, x) begin: DEMUX1_2_B_LOGIC
    if ((s == 0)) begin
        y0 = x;
        y1 = 0;
    end
    else begin
        y0 = 0;
        y1 = x;
    end
end

endmodule
```

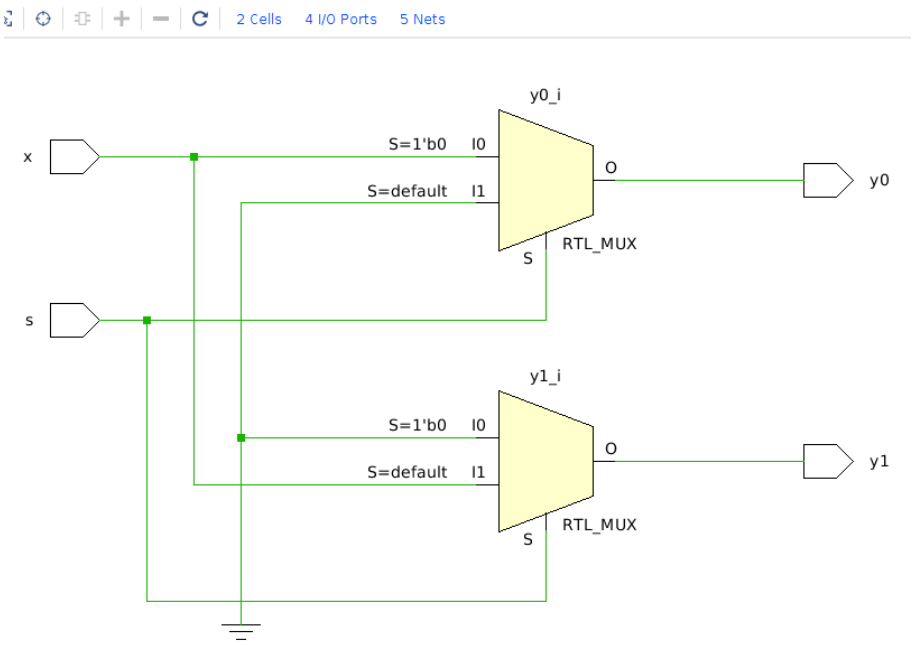


Figure 13: DEMUX1_2_B RTL schematic; Xilinx Vivado 2017.4

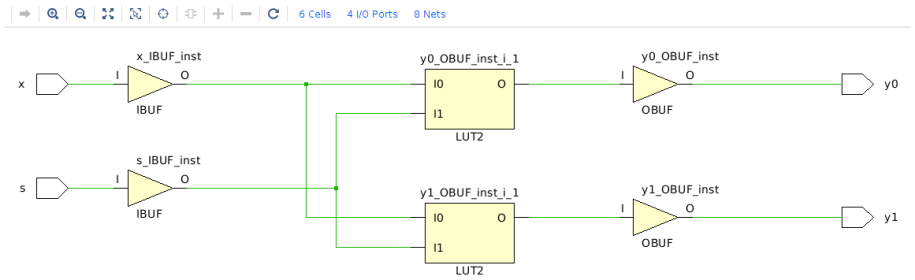


Figure 14: DEMUX1_2_B Synthesized Schematic; Xilinx Vivado 2017.4

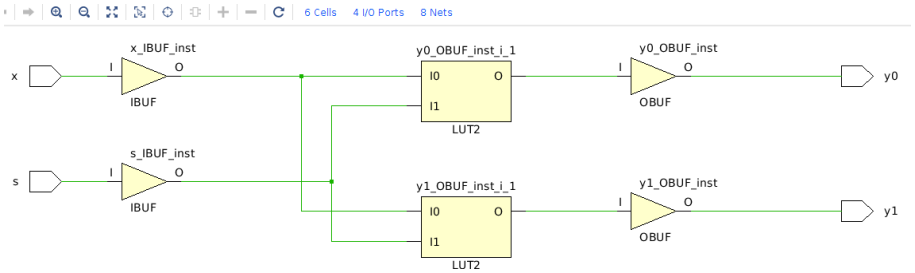


Figure 15: DEMUX1_2_B Implementated Schematic; Xilinx Vivado 2017.4

7.4 myHDL to Verilog Testbench

In [48]:
```
#create BitVectors
xTVs=intbv(int(''.join(xTVs.astype(str)), 2))[TestLen:]
sTVs=intbv(int(''.join(sTVs.astype(str)), 2))[TestLen:]

xTVs, bin(xTVs), sTVs, bin(sTVs)
```

Out[48]:
```
(intbv(5479), '1010101100111', intbv(3830), '111011110110')
```

In [49]:

```
@block
def DEMUX1_2_B_TBV():
    """
    myHDL -> testbench for module `DEMUX1_2_B`
    """

    x=Signal(bool(0))
    s=Signal(bool(0))
    y0=Signal(bool(0))
    y1=Signal(bool(0))

    @always_comb
    def print_data():
        print(x, s, y0, y1)

    #Test Signal Bit Vectors

    xTV=Signal(xTVs)
    sTV=Signal(sTVs)

    DUT=DEMUX1_2_B(x, s, y0, y1)


    @instance
    def stimules():
        for i in range(TestLen):
            x.next=int(xTV[i])
            s.next=int(sTV[i])

            yield delay(1)

        raise StopSimulation()

    return instances()

TB=DEMUX1_2_B_TBV()
TB.convert(hdl="Verilog", initial_values=True)
VerilogTextReader('DEMUX1_2_B_TBV');
```

```
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
<class 'myhdl._Signal._Signal'> <class '_ast.Name'>
***Verilog modual from DEMUX1_2_B_TBV.v***

 // File: DEMUX1_2_B_TBV.v
// Generated by MyHDL 0.10
// Date: Sun Sep 23 18:24:23 2018


`timescale 1ns/10ps

module DEMUX1_2_B_TBV (

);
// myHDL -> testbench for module `DEMUX1_2_B`
```

```
reg x = 0;
reg s = 0;
reg y0 = 0;
reg y1 = 0;
wire [13:0] xTV;
wire [13:0] sTV;

assign xTV = 14'd5479;
assign sTV = 14'd3830;


always @(x, y0, s, y1) begin: DEMUX1_2_B_TBV_PRINT_DATA
    $write("%h", x);
    $write(" ");
    $write("%h", s);
    $write(" ");
    $write("%h", y0);
    $write(" ");
    $write("%h", y1);
    $write("\n");
end


always @(s, x) begin: DEMUX1_2_B_TBV_DEMUX1_2_B0_0_LOGIC
    if ((s == 0)) begin
        y0 = x;
        y1 = 0;
    end
    else begin
        y0 = 0;
        y1 = x;
    end
end


initial begin: DEMUX1_2_B_TBV_STIMULES
    integer i;
    for (i=0; i<14; i=i+1) begin
        x <= xTV[i];
        s <= sTV[i];
        # 1;
    end
    $finish;
end

endmodule


/home/iridium/anaconda3/lib/python3.6/site-packages/myhdl/conversion/_t
oVerilog.py:349: ToVerilogWarning: Signal is not driven: xTV
  category=ToVerilogWarning
/home/iridium/anaconda3/lib/python3.6/site-packages/myhdl/conversion/_t
oVerilog.py:349: ToVerilogWarning: Signal is not driven: sTV
  category=ToVerilogWarning
```

7.5 PYNQ-Z1 Deployment

7.5.1 Board Circuit

See Board Circuit for "1 Channel Input: 2 Channel Output demultiplexer in Gate Level Logic"

7.5.2 Board Constraint

uses same 'DEMUX1_2.xdc' as "1 Channel Input: 2 Channel Output demultiplexer in Gate Level Logic"

7.5.3 Video of Deployment

DEMUX1_2_B on PYNQ-Z1 (YouTube (https://www.youtube.com/watch?v=Ux0olpi2ppI))

8 1:4 DEMUX via Behavioral if-elif-else

8.1 myHDL Module

In [50]:

```
@block
def DEMUX1_4_B(x, s0, s1, y0, y1, y2, y3):
    """
    1:4 DEMUX written via behaviorial

    Inputs:
        x(bool): input feed
        s0(bool): channel select 0
        s1(bool): channel select 1

    Outputs:
        y0(bool): ouput channel 0
        y1(bool): ouput channel 1
        y2(bool): ouput channel 2
        y3(bool): ouput channel 3
    """

    @always_comb
    def logic():
        if s0==0 and s1==0:
            y0.next=x; y1.next=0
            y2.next=0; y3.next=0

        elif s0==1 and s1==0:
            y0.next=0; y1.next=x
            y2.next=0; y3.next=0

        elif s0==0 and s1==1:
            y0.next=0; y1.next=0
            y2.next=x; y3.next=0

        else:
            y0.next=0; y1.next=0
            y2.next=0; y3.next=x

    return instances()
```

8.2 myHDL Testing

In [51]:
```
TestLen=10
SystmaticVals=list(itertools.product([0,1], repeat=3))

xTVs=np.array([i[2] for i in SystmaticVals]).astype(int)
np.random.seed(15)
xTVs=np.append(xTVs, np.random.randint(0,2, TestLen)).astype(int)

s0TVs=np.array([i[1] for i in SystmaticVals]).astype(int)
#the random genrator must have a differint seed beween each generation
#call in order to produce differint values for each call
np.random.seed(16)
s0TVs=np.append(s0TVs, np.random.randint(0,2, TestLen)).astype(int)

s1TVs=np.array([i[0] for i in SystmaticVals]).astype(int)
#the random genrator must have a differint seed beween each generation
#call in order to produce differint values for each call
np.random.seed(17)
s1TVs=np.append(s1TVs, np.random.randint(0,2, TestLen)).astype(int)

TestLen=len(xTVs)
SystmaticVals, xTVs, s0TVs, s1TVs
```

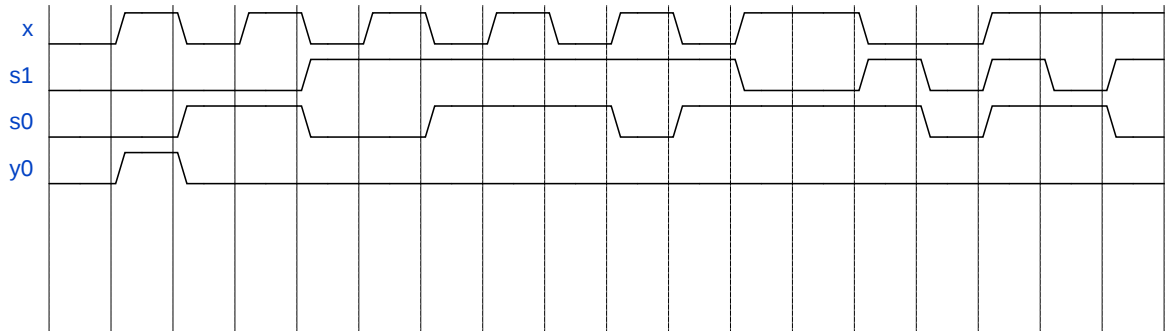
Out[51]:
```
([(0, 0, 0),
  (0, 0, 1),
  (0, 1, 0),
  (0, 1, 1),
  (1, 0, 0),
  (1, 0, 1),
  (1, 1, 0),
  (1, 1, 1)],
 array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1]),
 array([0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0]),
 array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1]))
```

In [52]:

```
Peeker.clear()
x=Signal(bool(0)); Peeker(x, 'x')
s0=Signal(bool(0)); Peeker(s0, 's0')
s1=Signal(bool(0)); Peeker(s1, 's1')
y0=Signal(bool(0)); Peeker(y0, 'y0')
y1=Signal(bool(0)); Peeker(y1, 'y1')
y2=Signal(bool(0)); Peeker(y2, 'y2')
y3=Signal(bool(0)); Peeker(y3, 'y3')


DUT=DEMUX1_4_B(x, s0, s1, y0, y1, y2, y3)

def DEMUX1_4_B_TB():
    """
    myHDL only testbench for module `DEMUX1_4_Combo`
    """

    @instance
    def stimules():
        for i in range(TestLen):
            x.next=int(xTVs[i])
            s0.next=int(s0TVs[i])
            s1.next=int(s1TVs[i])


            yield delay(1)

        raise StopSimulation()

    return instances()

sim=Simulation(DUT, DEMUX1_4_B_TB(), *Peeker.instances()).run()
```

In [53]:

```
Peeker.to_wavedrom('x', 's1', 's0', 'y0', 'y1', 'y2', 'y3')
```



In [54]:
```
DEMUX1_4_BData=Peeker.to_dataframe()
DEMUX1_4_BData=DEMUX1_4_BData[['x', 's1', 's0', 'y0', 'y1', 'y2', 'y3']]
DEMUX1_4_BData
```

Out[54]:

	x	s1	s0	y0	y1	y2	y3
0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0
2	0	0	1	0	0	0	0
3	1	0	1	0	1	0	0
4	0	1	0	0	0	0	0
5	1	1	0	0	0	1	0
6	0	1	1	0	0	0	0
7	1	1	1	0	0	0	1
8	0	1	1	0	0	0	0
9	1	1	0	0	0	1	0
10	0	1	1	0	0	0	0
11	1	0	1	0	1	0	0
13	0	1	1	0	0	0	0
14	0	0	0	0	0	0	0
15	1	1	1	0	0	0	1
16	1	0	1	0	1	0	0
17	1	1	0	0	0	1	0

In [55]:
```
Test=DEMUX1_4_BData==DEMUX1_4_ComboData[['x', 's1', 's0', 'y0', 'y1', '
Test=Test.all().all()
print(f'DEMUX1_4_B equivlinet to DEMUX1_4_Combo: {Test}')
```

```
DEMUX1_4_B equivlinet to DEMUX1_4_Combo: True
```

8.3 Verilog Conversion

In [56]:
```
DUT.convert()
VerilogTextReader('DEMUX1_4_B');
```

```
***Verilog modual from DEMUX1_4_B.v***

 // File: DEMUX1_4_B.v
// Generated by MyHDL 0.10
// Date: Sun Sep 23 18:24:25 2018


`timescale 1ns/10ps

module DEMUX1_4_B (
    x,
    s0,
    s1,
    y0,
    y1,
    y2,
    y3
);
// 1:4 DEMUX written via behaviorial
//
// Inputs:
//     x(bool): input feed
//     s0(bool): channel select 0
//     s1(bool): channel select 1
//
// Outputs:
//     y0(bool): ouput channel 0
//     y1(bool): ouput channel 1
//     y2(bool): ouput channel 2
//     y3(bool): ouput channel 3

input x;
input s0;
input s1;
output y0;
reg y0;
output y1;
reg y1;
output y2;
reg y2;
output y3;
reg y3;




always @(x, s1, s0) begin: DEMUX1_4_B_LOGIC
    if (((s0 == 0) && (s1 == 0))) begin
        y0 = x;
        y1 = 0;
        y2 = 0;
        y3 = 0;
    end
    else if (((s0 == 1) && (s1 == 0))) begin
```

```
            y0 = 0;
            y1 = x;
            y2 = 0;
            y3 = 0;
        end
        else if (((s0 == 0) && (s1 == 1))) begin
            y0 = 0;
            y1 = 0;
            y2 = x;
            y3 = 0;
        end
        else begin
            y0 = 0;
            y1 = 0;
            y2 = 0;
            y3 = x;
        end
end

endmodule
```

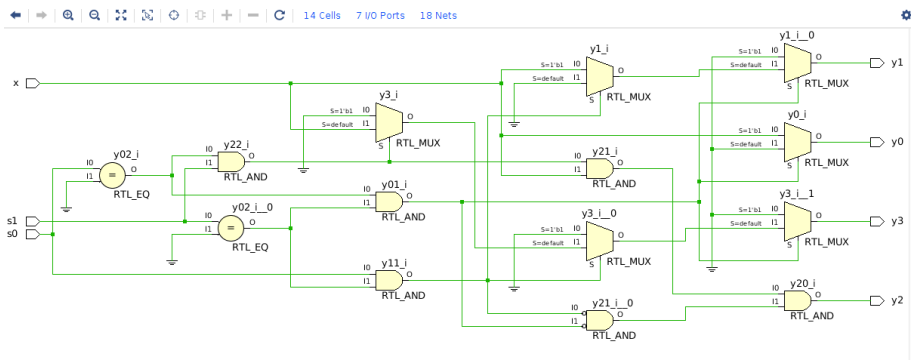


Figure 16: DEMUX1_4_B RTL schematic; Xilinx Vivado 2017.4

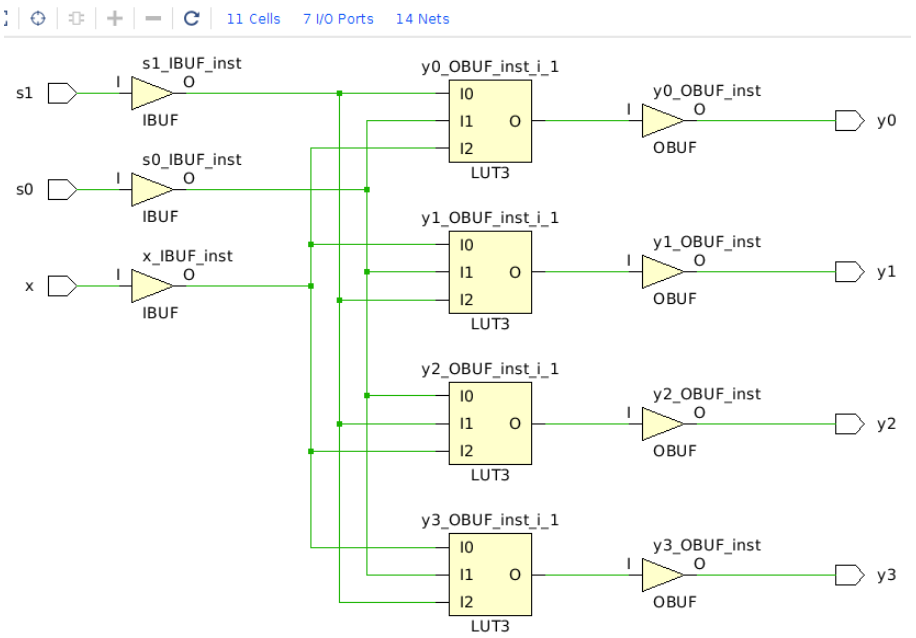


Figure 17: DEMUX1_4_B Synthesized Schematic; Xilinx Vivado 2017.4

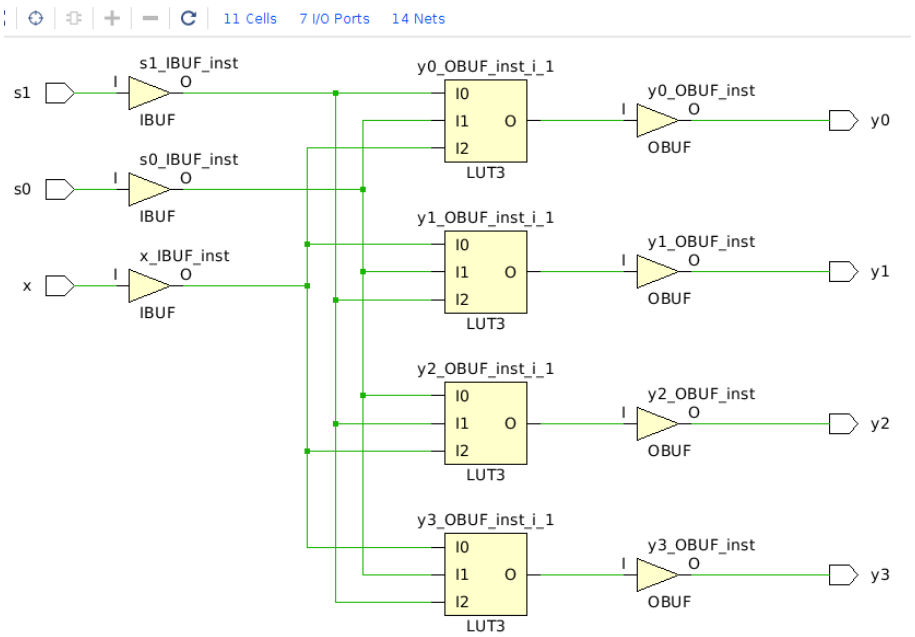


Figure 18: DEMUX1_4_B Implementated Schematic; Xilinx Vivado 2017.4

8.4 myHDL to Verilog Testbench

In [57]:
```
#create BitVectors
xTVs=intbv(int(''.join(xTVs.astype(str)), 2))[TestLen:]
s0TVs=intbv(int(''.join(s0TVs.astype(str)), 2))[TestLen:]
s1TVs=intbv(int(''.join(s1TVs.astype(str)), 2))[TestLen:]


xTVs, bin(xTVs), s0TVs, bin(s0TVs), s1TVs, bin(s1TVs)
```

Out[57]:
```
(intbv(87399),
 '10101010101100111',
 intbv(52982),
 '1100111011110110',
 intbv(16277),
 '11111110010101')
```

In [58]:

```
@block
def DEMUX1_4_B_TBV():
    """
    myHDL -> testbench for module `DEMUX1_4_B`
    """

    x=Signal(bool(0))
    s0=Signal(bool(0))
    s1=Signal(bool(0))
    y0=Signal(bool(0))
    y1=Signal(bool(0))
    y2=Signal(bool(0))
    y3=Signal(bool(0))

    @always_comb
    def print_data():
        print(x, s0, s1, y0, y1, y2, y3)

    #Test Signal Bit Vectors

    xTV=Signal(xTVs)
    s0TV=Signal(s0TVs)
    s1TV=Signal(s1TVs)

    DUT=DEMUX1_4_B(x, s0, s1, y0, y1, y2, y3)


    @instance
    def stimules():
        for i in range(TestLen):
            x.next=int(xTV[i])
            s0.next=int(s0TV[i])
            s1.next=int(s1TV[i])

            yield delay(1)

        raise StopSimulation()

    return instances()

TB=DEMUX1_4_B_TBV()
TB.convert(hdl="Verilog", initial_values=True)
VerilogTextReader('DEMUX1_4_B_TBV');
```

```
initial begin: DEMUX1_4_B_TBV_STIMULES
    integer i;
    for (i=0; i<18; i=i+1) begin
        x <= xTV[i];
        s0 <= s0TV[i];
        s1 <= s1TV[i];
        # 1;
    end
    $finish;
end

endmodule
```

```
/home/iridium/anaconda3/lib/python3.6/site-packages/myhdl/conversion/
_toVerilog.py:349: ToVerilogWarning: Signal is not driven: xTV
  category=ToVerilogWarning
/home/iridium/anaconda3/lib/python3.6/site-packages/myhdl/conversion/
 toVerilog.py:349: ToVerilogWarning: Signal is not driven: s0TV
```

8.5 PYNQ-Z1 Deployment

8.5.1 Board Circuit

See Board Circuit for "1 Channel Input:4 Channel Output demultiplexer in Gate Level Logic"

8.5.2 Board Constraint

uses same 'DEMUX1_4.xdc' as "# 1 Channel Input:4 Channel Output demultiplexer in Gate Level Logic"

8.5.3 Video of Deployment

DEMUX1_4_B on PYNQ-Z1 (YouTube (https://www.youtube.com/watch?v=MezWijKoDuk))

9 Demultiplexer 1:4 Behavioral via Bitvectors

9.1 myHDL Module

In [59]:
```
@block
def DEMUX1_4_BV(x, S, Y):
    """
    1:4 DEMUX written via behaviorial with
    bit vectors

    Inputs:
        x(bool): input feed
        S(2bit vector): channel select bitvector;
            min=0, max=3

    Outputs:
        Y(4bit vector): ouput channel bitvector;
            values min=0, max=15; allowed is: 0,1,2,4,8
            in this application

    """

    @always_comb
    def logic():
        #here concat is used to build up the word
        #from the x input
        if S==0:
            Y.next=concat(intbv(0)[3:], x); '0001'
        elif S==1:
            Y.next=concat(intbv(0)[2:], x, intbv(0)[1:]); '0010'
        elif S==2:
            Y.next=concat(intbv(0)[1:], x, intbv(0)[2:]); '0100'
        else:
            Y.next=concat(x, intbv(0)[3:]); '1000'

    return instances()
```

9.2 myHDL Testing

In [60]:
```
xTVs=np.array([0,1])
xTVs=np.append(xTVs, np.random.randint(0,2,6)).astype(int)
TestLen=len(xTVs)

np.random.seed(12)
STVs=np.arange(0,4)
STVs=np.append(STVs, np.random.randint(0,4, 5))
TestLen, xTVs, STVs
```

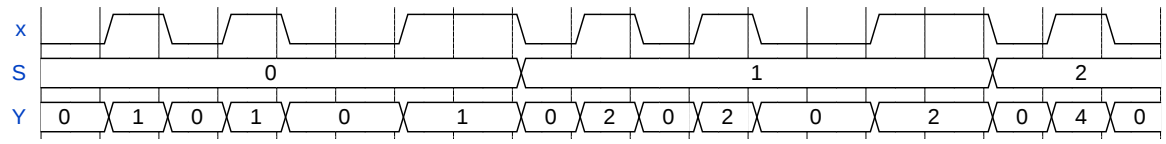
Out[60]:
```
(8, array([0, 1, 0, 1, 0, 0, 1, 1]), array([0, 1, 2, 3, 3, 3, 2, 1,
1]))
```

In [61]:

```
Peeker.clear()
x=Signal(bool(0)); Peeker(x, 'x')
S=Signal(intbv(0)[2:]); Peeker(S, 'S')
Y=Signal(intbv(0)[4:]); Peeker(Y, 'Y')

DUT=DEMUX1_4_BV(x, S, Y)

def DEMUX1_4_BV_TB():

    @instance
    def stimules():
        for i in STVs:
            for j in xTVs:
                S.next=int(i)
                x.next=int(j)
                yield delay(1)

        raise StopSimulation()

    return instances()

sim=Simulation(DUT, DEMUX1_4_BV_TB(), *Peeker.instances()).run()
```

In [63]:

```
Peeker.to_wavedrom('x', 'S', 'Y', start_time=0, stop_time=2*TestLen+2)
```



In [66]:

```
DEMUX1_4_BVData=Peeker.to_dataframe()
DEMUX1_4_BVData=DEMUX1_4_BVData[['x', 'S', 'Y']]
DEMUX1_4_BVData
```

Out[66]:

	x	S	Y
0	0	0	0
1	1	0	1
2	0	0	0
3	1	0	1
4	0	0	0
6	1	0	1
8	0	1	0
9	1	1	2
10	0	1	0
11	1	1	2
12	0	1	0

In [67]:

```
DEMUX1_4_BVData['y0']=None; DEMUX1_4_BVData['y1']=None; DEMUX1_4_BVData
DEMUX1_4_BVData[['y3', 'y2', 'y1', 'y0']]=DEMUX1_4_BVData[['Y']].apply(

DEMUX1_4_BVData['s0']=None; DEMUX1_4_BVData['s1']=None
DEMUX1_4_BVData[['s1', 's0']]=DEMUX1_4_BVData[['S']].apply(lambda bv: [

DEMUX1_4_BVData=DEMUX1_4_BVData[['x', 'S', 's0', 's1', 'Y', 'y3', 'y2',
DEMUX1_4_BVData
```

46	1	3	1	1	8	1	0	0	0
48	0	2	0	1	0	0	0	0	0
49	1	2	0	1	4	0	1	0	0
50	0	2	0	1	0	0	0	0	0
51	1	2	0	1	4	0	1	0	0
52	0	2	0	1	0	0	0	0	0
54	1	2	0	1	4	0	1	0	0
56	0	1	1	0	0	0	0	0	0
57	1	1	1	0	2	0	0	1	0
58	0	1	1	0	0	0	0	0	0
59	1	1	1	0	2	0	0	1	0
60	0	1	1	0	0	0	0	0	0
62	1	1	1	0	2	0	0	1	0

In [70]:
```
DEMUX1_4_BVData['y0Ref']=DEMUX1_4_BVData.apply(lambda row:y14_0EqN(row[
DEMUX1_4_BVData['y1Ref']=DEMUX1_4_BVData.apply(lambda row:y14_1EqN(row[
DEMUX1_4_BVData['y2Ref']=DEMUX1_4_BVData.apply(lambda row:y14_2EqN(row[
DEMUX1_4_BVData['y3Ref']=DEMUX1_4_BVData.apply(lambda row:y14_3EqN(row[

DEMUX1_4_BVData
```

16	0	2	0	1	0	0	0	0	0	0	0	0
17	1	2	0	1	4	0	1	0	0	0	1	0
18	0	2	0	1	0	0	0	0	0	0	0	0
19	1	2	0	1	4	0	1	0	0	0	1	0
20	0	2	0	1	0	0	0	0	0	0	0	0
22	1	2	0	1	4	0	1	0	0	0	1	0
24	0	3	1	1	0	0	0	0	0	0	0	0
25	1	3	1	1	8	1	0	0	0	0	0	1
26	0	3	1	1	0	0	0	0	0	0	0	0
27	1	3	1	1	8	1	0	0	0	0	0	1
28	0	3	1	1	0	0	0	0	0	0	0	0
30	1	3	1	1	8	1	0	0	0	0	0	1
32	0	3	1	1	0	0	0	0	0	0	0	0

In [84]:
```
X1_4_BVData[['y0', 'y1', 'y2', 'y3']].sort_index(inplace=True)==DEMUX1_4_
odule `DEMUX1_4_BVData` works as exspected: {Test}')
```

```
Module `DEMUX1_4_BVData` works as exspected: True
```

9.3 Verilog Conversion

In [85]:
```
DUT.convert()
VerilogTextReader('DEMUX1_4_BV');
```

```
***Verilog modual from DEMUX1_4_BV.v***

 // File: DEMUX1_4_BV.v
// Generated by MyHDL 0.10
// Date: Sun Sep 23 18:48:43 2018


`timescale 1ns/10ps

module DEMUX1_4_BV (
    x,
    S,
    Y
);
// 1:4 DEMUX written via behaviorial with
// bit vectors
//
// Inputs:
//     x(bool): input feed
//     S(2bit vector): channel select bitvector;
//         min=0, max=3
//
// Outputs:
//     Y(4bit vector): ouput channel bitvector;
//         values min=0, max=15; allowed is: 0,1,2,4,8
//         in this application
//

input x;
input [1:0] S;
output [3:0] Y;
reg [3:0] Y;




always @(x, S) begin: DEMUX1_4_BV_LOGIC
    case (S)
        'h0: begin
            Y = {3'h0, x};
            // 0001
        end
        'h1: begin
            Y = {2'h0, x, 1'h0};
            // 0010
        end
        'h2: begin
            Y = {1'h0, x, 2'h0};
            // 0100
        end
        default: begin
            Y = {x, 3'h0};
            // 1000
        end
```

```
    endcase
end

endmodule
```

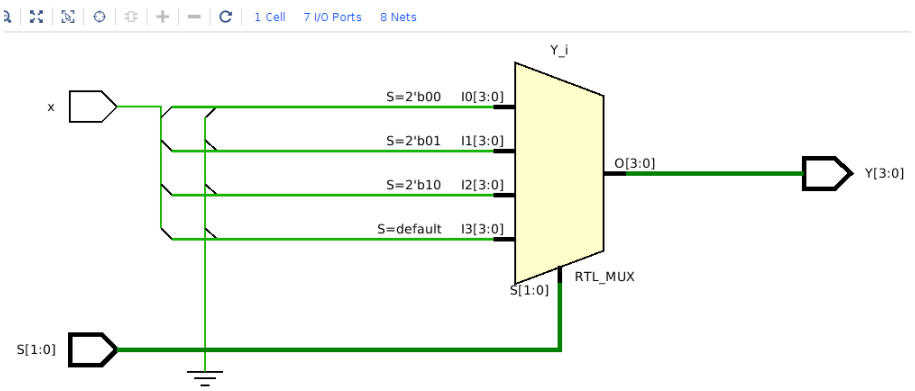


Figure 19: DEMUX1_4_BV RTL schematic; Xilinx Vivado 2017.4

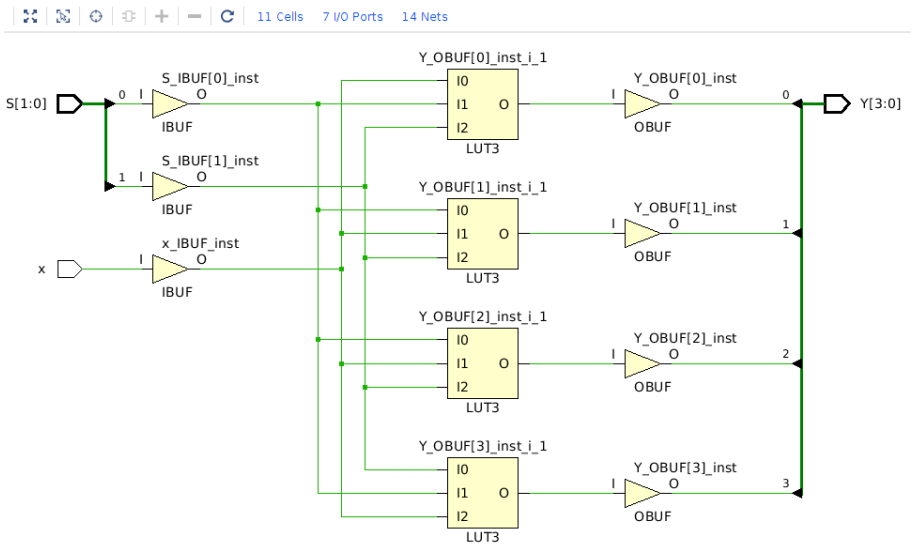


Figure 20: DEMUX1_4_BV Synthesized Schematic; Xilinx Vivado 2017.4

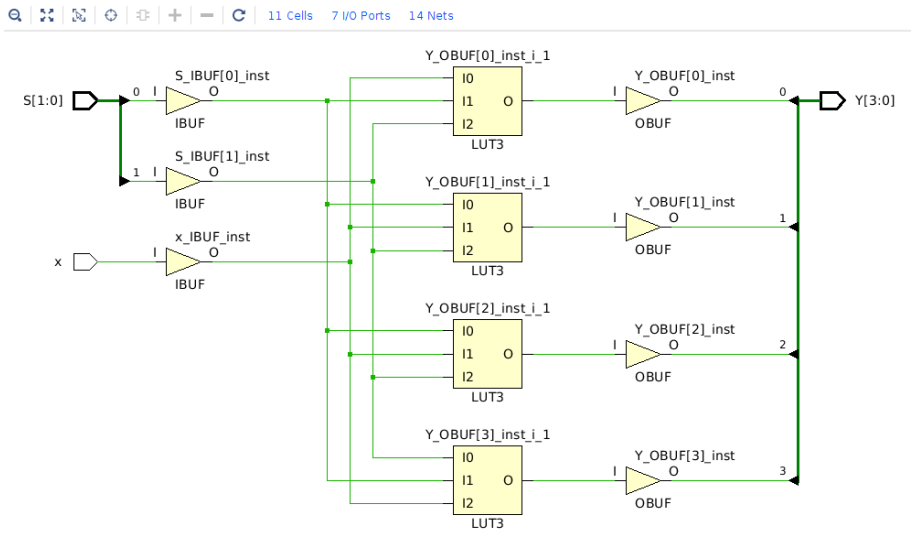


Figure 21: DEMUX1_4_BV Implementated Schematic; Xilinx Vivado 2017.4

9.4 myHDL to Verilog Testbench

(To Do!)

9.5 PYNQ-Z1 Board Deployment

9.5.1 Board Circuit

9.5.2 Board Constraints

In [86]:
```
ConstraintXDCTextReader('DEMUX1_4_BV');
```

```
***Constraint file from DEMUX1_4_BV.xdc***

 ## Switches
set_property -dict {PACKAGE_PIN M20 IOSTANDARD LVCMOS33} [get_ports {S
[0]}]; ##SW0
set_property -dict {PACKAGE_PIN M19 IOSTANDARD LVCMOS33} [get_ports {S
[1]}]; ##SW1

## Buttons

set_property -dict {PACKAGE_PIN L19 IOSTANDARD LVCMOS33} [get_ports
{x}]; ##BT3


## RGBLEDs
set_property -dict { PACKAGE_PIN L15   IOSTANDARD LVCMOS33 } [get_ports
{Y[0]}]; ##LD4_Blue
set_property -dict { PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports
{Y[1]}]; ##LD4_Green
set_property -dict { PACKAGE_PIN L14   IOSTANDARD LVCMOS33 } [get_ports
{Y[2]}]; ##LD5_Green
set_property -dict { PACKAGE_PIN M15   IOSTANDARD LVCMOS33 } [get_ports
{Y[3]}]; ##LD5_Red
```

9.5.3 Video of Deployment

DEMUX1_4_BV on PYNQ-Z1 (YouTube (https://www.youtube.com/watch?v=mVyTlkbJpKg))