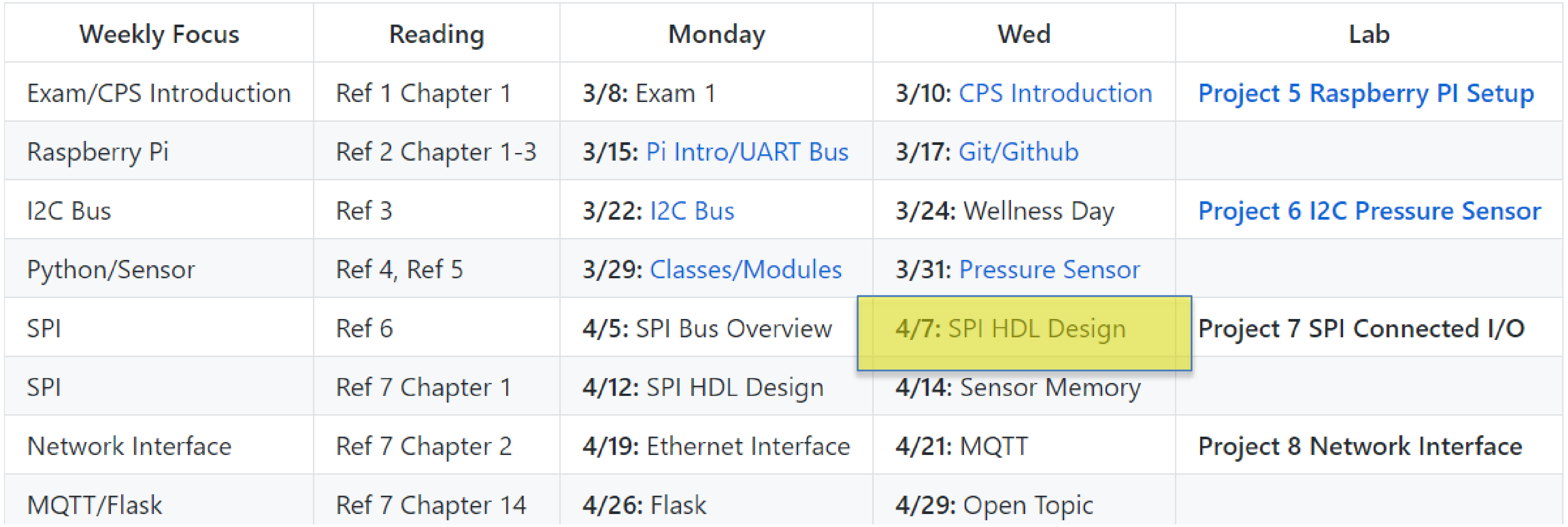
E210 Engineering Cyber-Physical Systems (Spring 2021)

SPI Peripheral (Basys3)

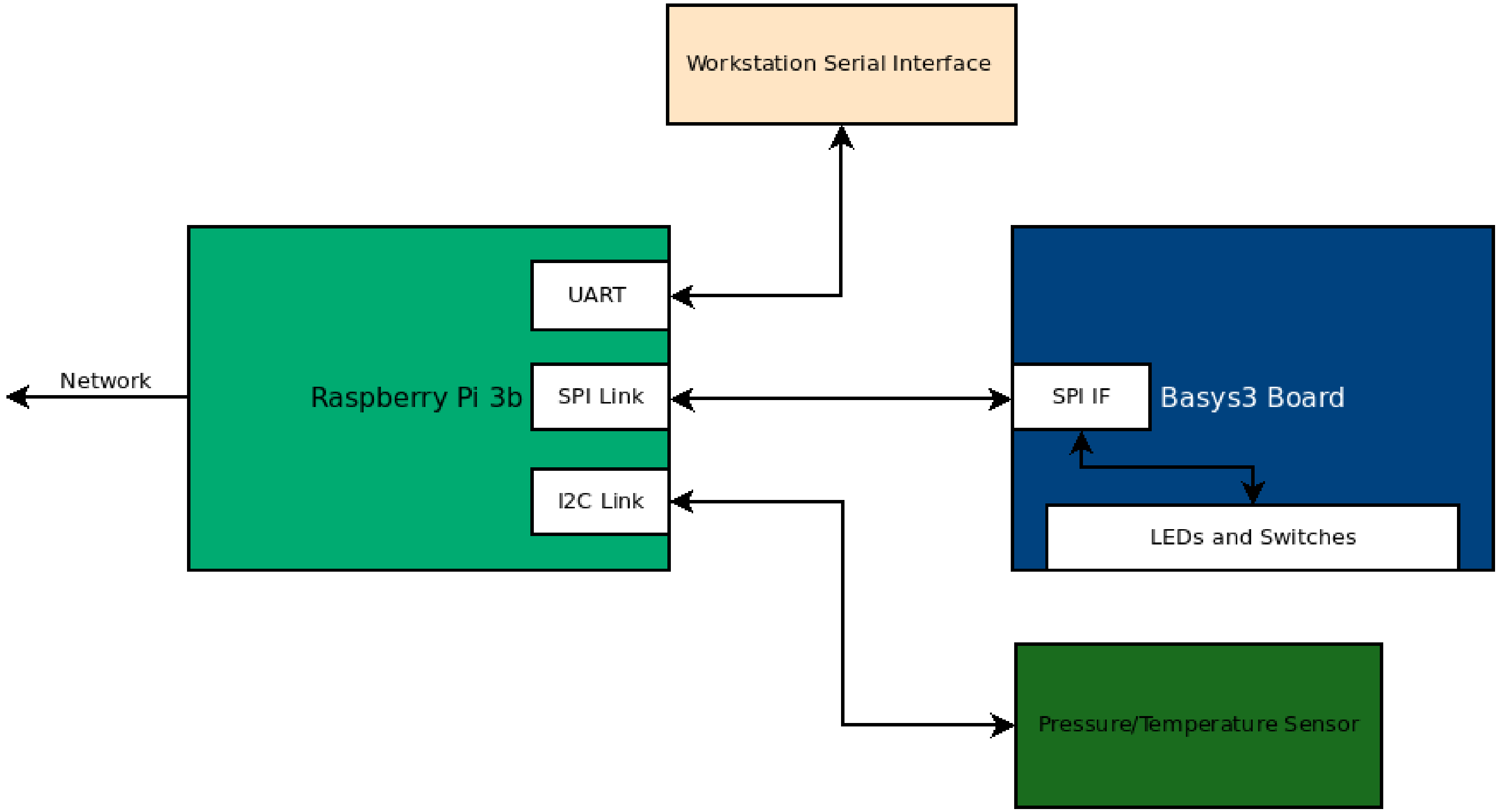
Bryce Himebaugh



Weekly Focus	Reading	Monday	Wed	Lab
Exam/CPS Introduction	Ref 1 Chapter 1	**3/8:** Exam 1	**3/10:** CPS Introduction	**Project 5 Raspberry PI Setup**
Raspberry Pi	Ref 2 Chapter 1-3	**3/15:** Pi Intro/UART Bus	**3/17:** Git/Github	
I2C Bus	Ref 3	**3/22:** I2C Bus	**3/24:** Wellness Day	**Project 6 I2C Pressure Sensor**
Python/Sensor	Ref 4, Ref 5	**3/29:** Classes/Modules	**3/31:** Pressure Sensor	
SPI	Ref 6	**4/5:** SPI Bus Overview	**4/7:** SPI HDL Design	**Project 7 SPI Connected I/O**
SPI	Ref 7 Chapter 1	**4/12:** SPI HDL Design	**4/14:** Sensor Memory	
Network Interface	Ref 7 Chapter 2	**4/19:** Ethernet Interface	**4/21:** MQTT	**Project 8 Network Interface**
MQTT/Flask	Ref 7 Chapter 14	**4/26:** Flask	**4/29:** Open Topic	

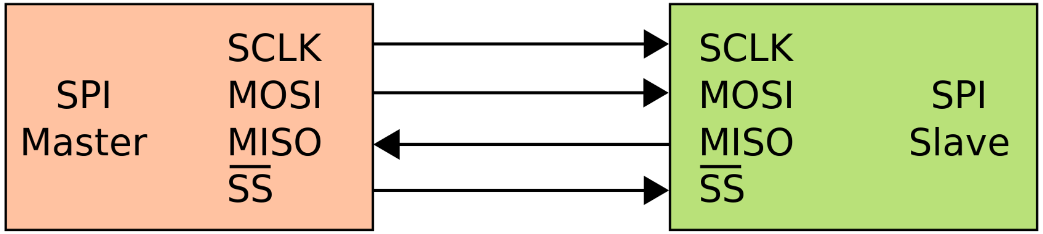
https://engr210.github.io/

Raspberry SPI Link



SPI Bus Review

Single Peripheral

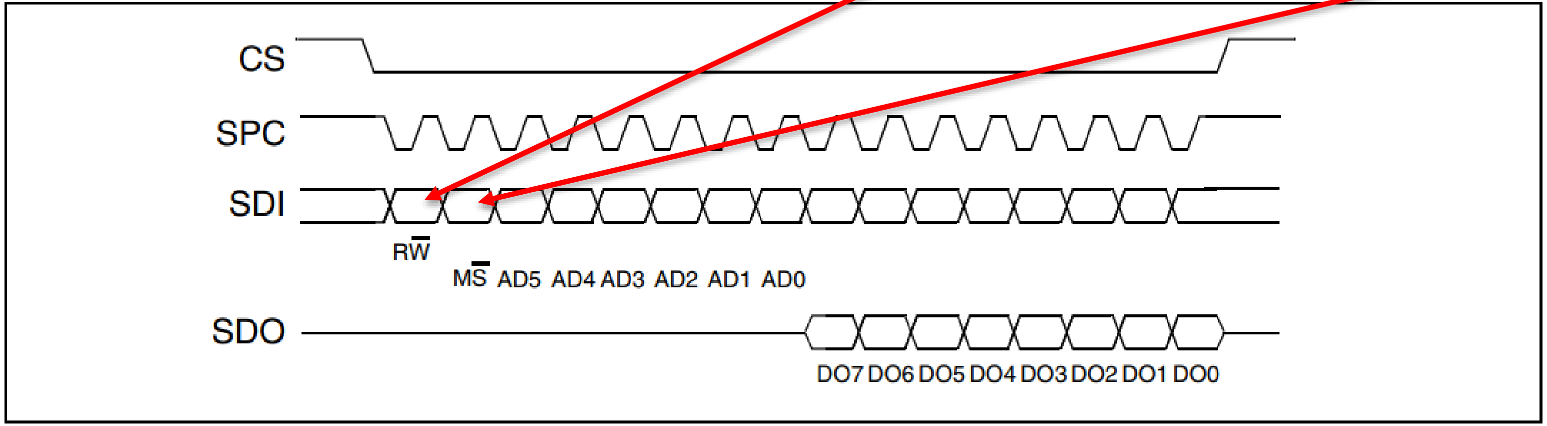


Reading

Set to 1, for reading

Auto-Increment Address

Figure 5. SPI read protocol



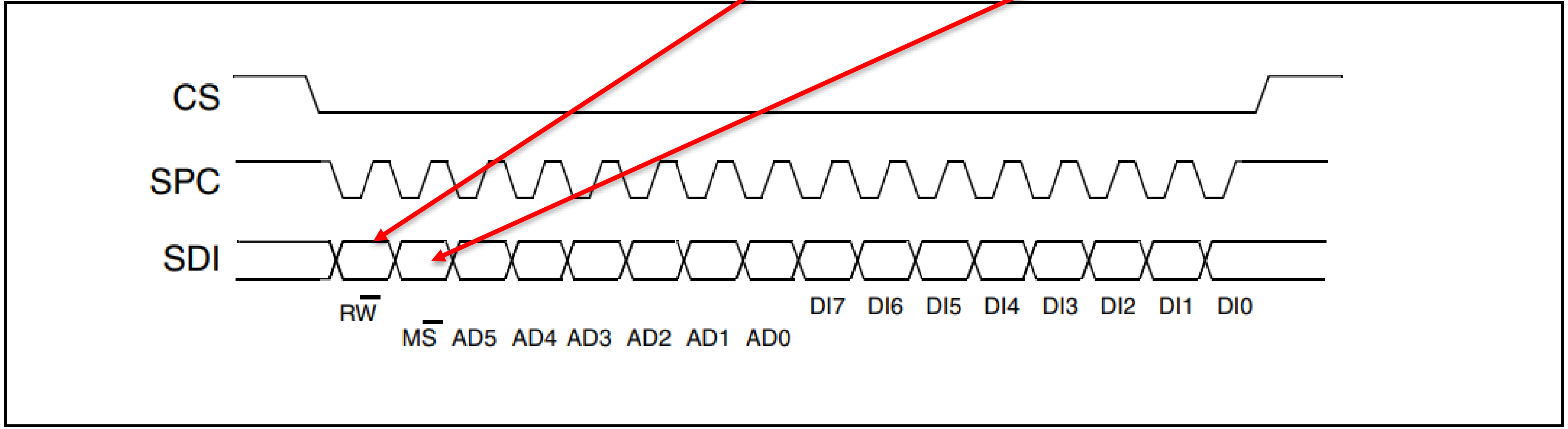
The SPI Read command is performed with 16 clock pulses. The multiple byte read command is performed adding blocks of 8 clock pulses at the previous one.

Writing

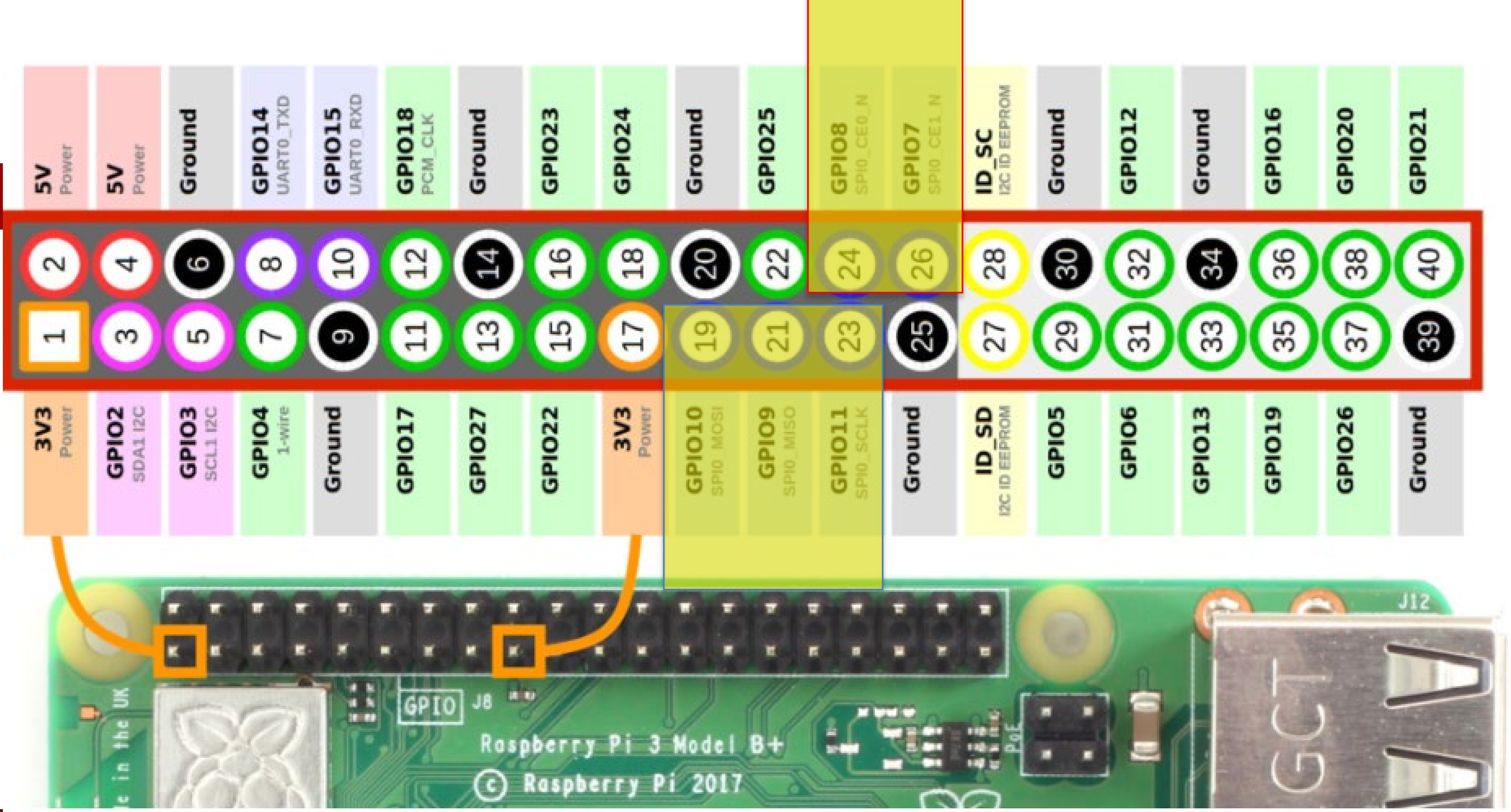
Set to 0, for writing

Multiple Register Write …

Figure 7. SPI write protocol



Raspberry Pi Python SPI


5V Power
5V Power
Ground
GPIO14 UART0_TXD
GPIO15 UART0_RXD
GPIO18 PCM_CLK
Ground
GPIO23
GPIO24
Ground
GPIO25
GPIO8 SPI0_CE0_N
GPIO7 SPI0_CE1_N
ID_SC I2C ID EEPROM
Ground
GPIO12
Ground
GPIO16
GPIO20
GPIO21
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
3V3 Power
GPIO2 SDA1 I2C
GPIO3 SCL1 I2C
GPIO4 1-wire
Ground
GPIO17
GPIO27
GPIO22
3V3 Power
GPIO10 SPI0_MOSI
GPIO9 SPI0_MISO
GPIO11 SPI0_SCLK
Ground
ID_SD I2C ID EEPROM
GPIO5
GPIO6
GPIO13
GPIO19
GPIO26
Ground
J12
GPIO
J8
Raspberry Pi 3 Model B+
© Raspberry Pi 2017
PoE
Made in the UK

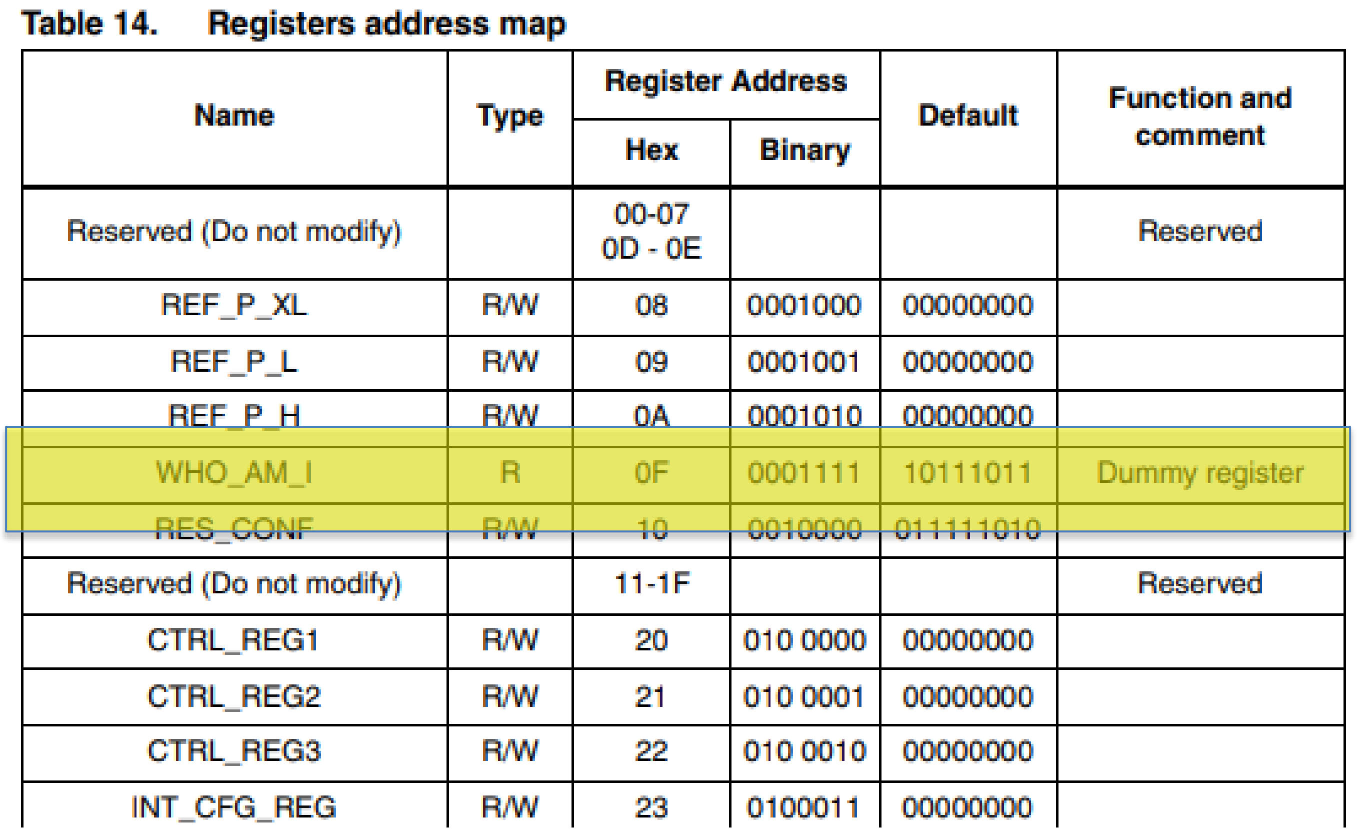
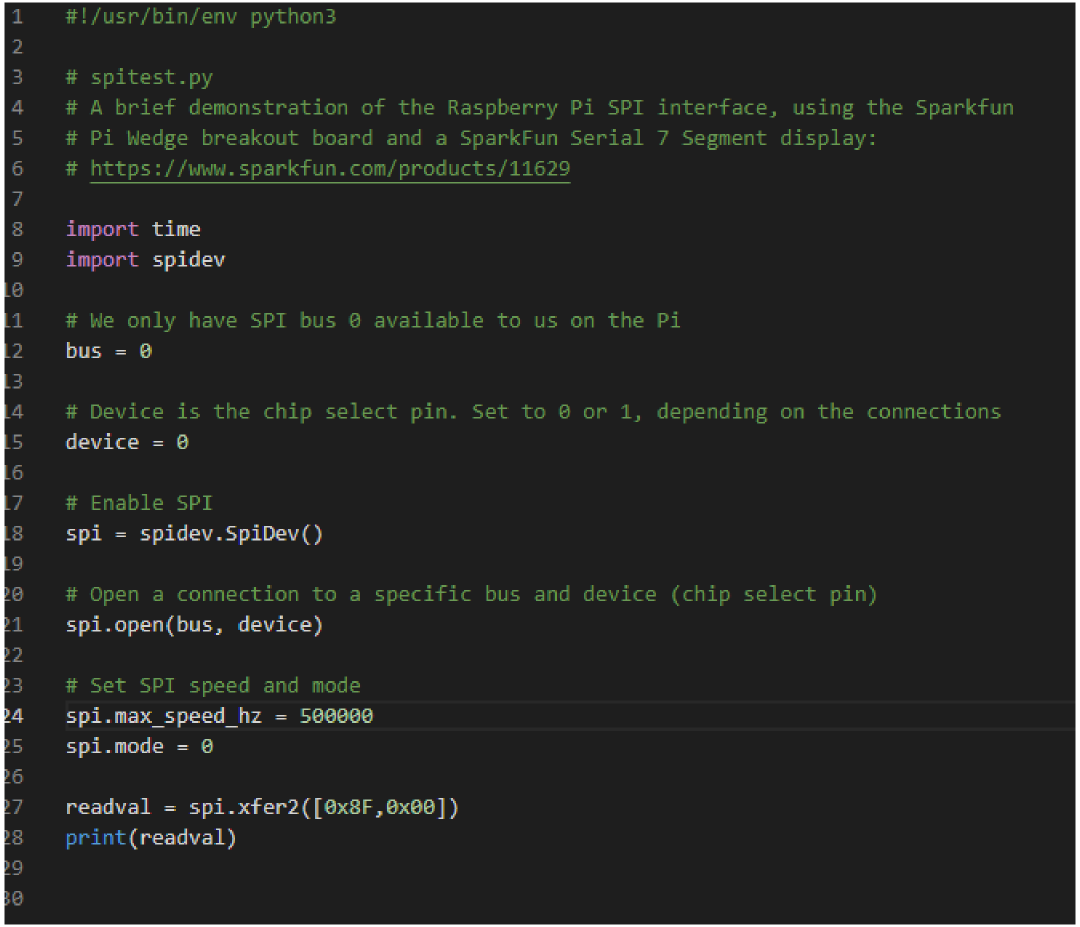
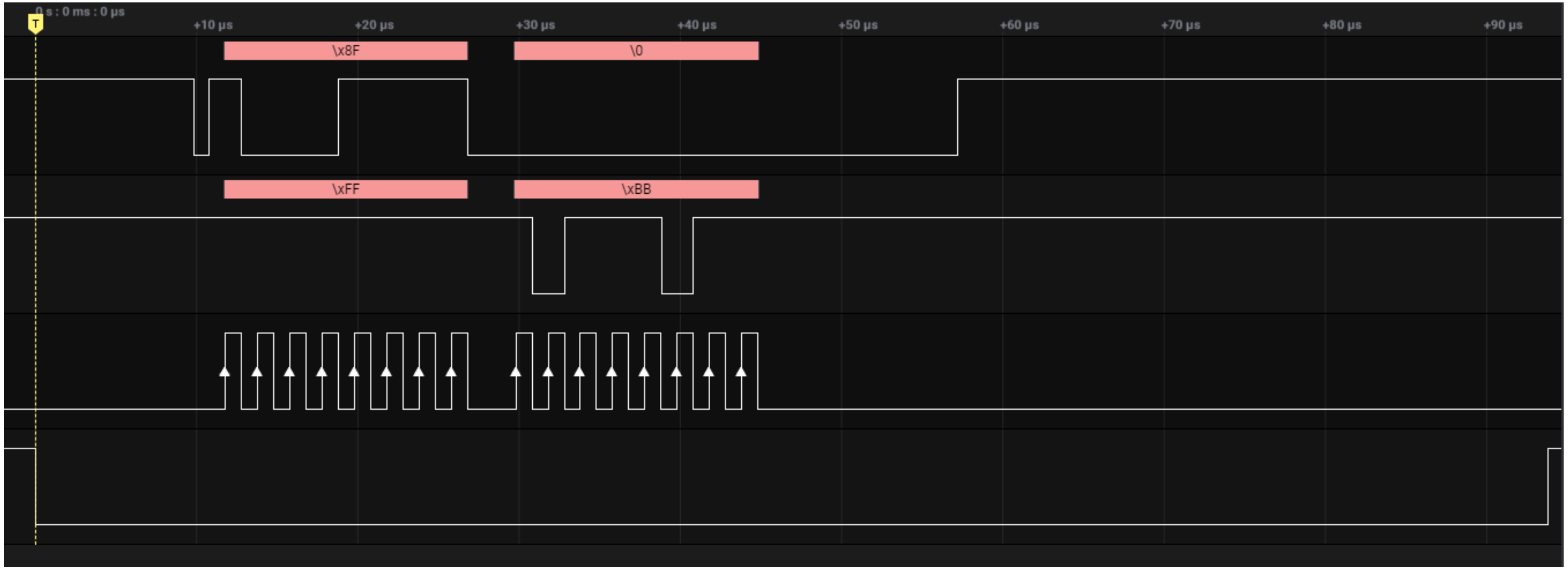


Table 14. Registers address map

Name	Type	Register Address		Default	Function and comment
		Hex	Binary		
Reserved (Do not modify)		00-07 0D - 0E			Reserved
REF_P_XL	R/W	08	0001000	00000000	
REF_P_L	R/W	09	0001001	00000000	
REF_P_H	R/W	0A	0001010	00000000	
WHO_AM_I	R	0F	0001111	10111011	Dummy register
RES_CONF	R/W	10	0010000	011111010	
Reserved (Do not modify)		11-1F			Reserved
CTRL_REG1	R/W	20	010 0000	00000000	
CTRL_REG2	R/W	21	010 0001	00000000	
CTRL_REG3	R/W	22	010 0010	00000000	
INT_CFG_REG	R/W	23	0100011	00000000	



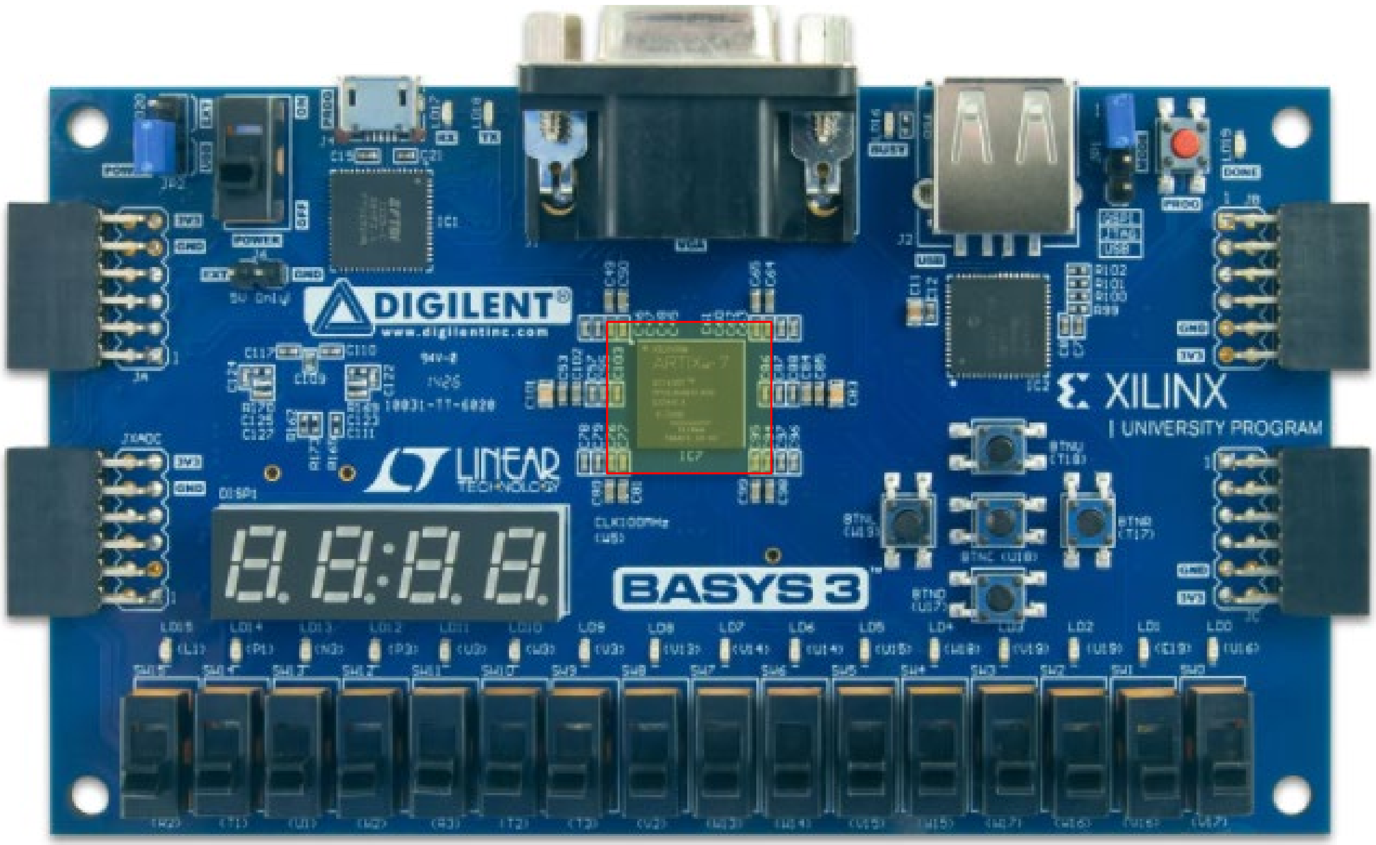
```python
#!/usr/bin/env python3

# spitest.py
# A brief demonstration of the Raspberry Pi SPI interface, using the Sparkfun
# Pi Wedge breakout board and a SparkFun Serial 7 Segment display:
# https://www.sparkfun.com/products/11629

import time
import spidev

# We only have SPI bus 0 available to us on the Pi
bus = 0

# Device is the chip select pin. Set to 0 or 1, depending on the connections
device = 0

# Enable SPI
spi = spidev.SpiDev()

# Open a connection to a specific bus and device (chip select pin)
spi.open(bus, device)

# Set SPI speed and mode
spi.max_speed_hz = 500000
spi.mode = 0

readval = spi.xfer2([0x8F,0x00])
print(readval)


```


0 s : 0 ms : 0 µs
T
+10 µs
+20 µs
+30 µs
+40 µs
+50 µs
+60 µs
+70 µs
+80 µs
+90 µs
\x8F
\0
\xFF
\xBB

Xilinx Artix 7 XC7A35T

Front View

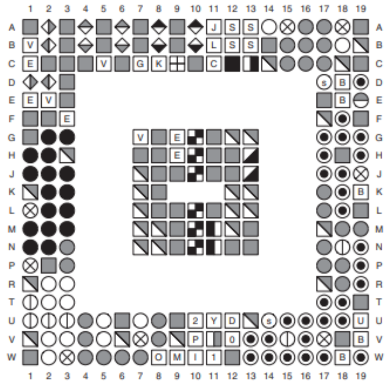
Back View


DIGILENT
www.digilentinc.com
ARTIX-7
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
CLK100MHz
BTNU
BTNL
BTNC
BTNR
BTND
POWER
PROG
DONE
VGA
USB

Front View

CPG236 BGA Package

CP236 and CPG236 Packages—XC7A15T, XC7A35T, and XC7A50T
CPG236 Package (only)—XA7A15T, XA7A35T, and XA7A50T



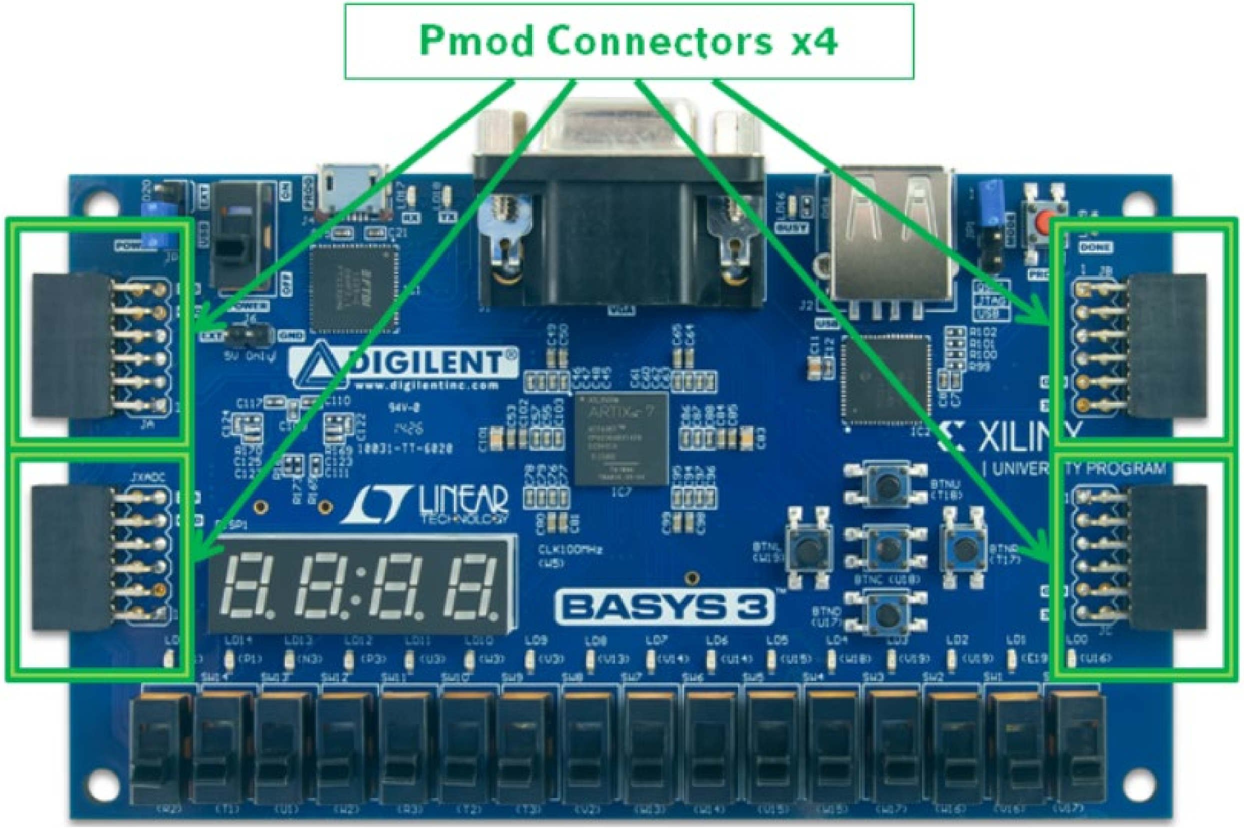
User I/O Pins	
○	IO_LXXY_#
ⓢ	IO_XX_#
Multi–Function Pins	
B	ADV_B
B	FCS_B
B	FOE_B
B	MOSI
B	FWE_B
B	DOUT_CSO_B
B	CSI_B
B	PUDC_B
U	RDWR_B
r	RS0–RS1
●	AD0P/AD0N–AD15P/AD15N
⊖	EMCCLK
⊕	VRN
⊖	VRP
⊗	VREF
⊙	D00–D31
◓	A00–A28
⦶	DQS
●	MRCC
●	SRCC

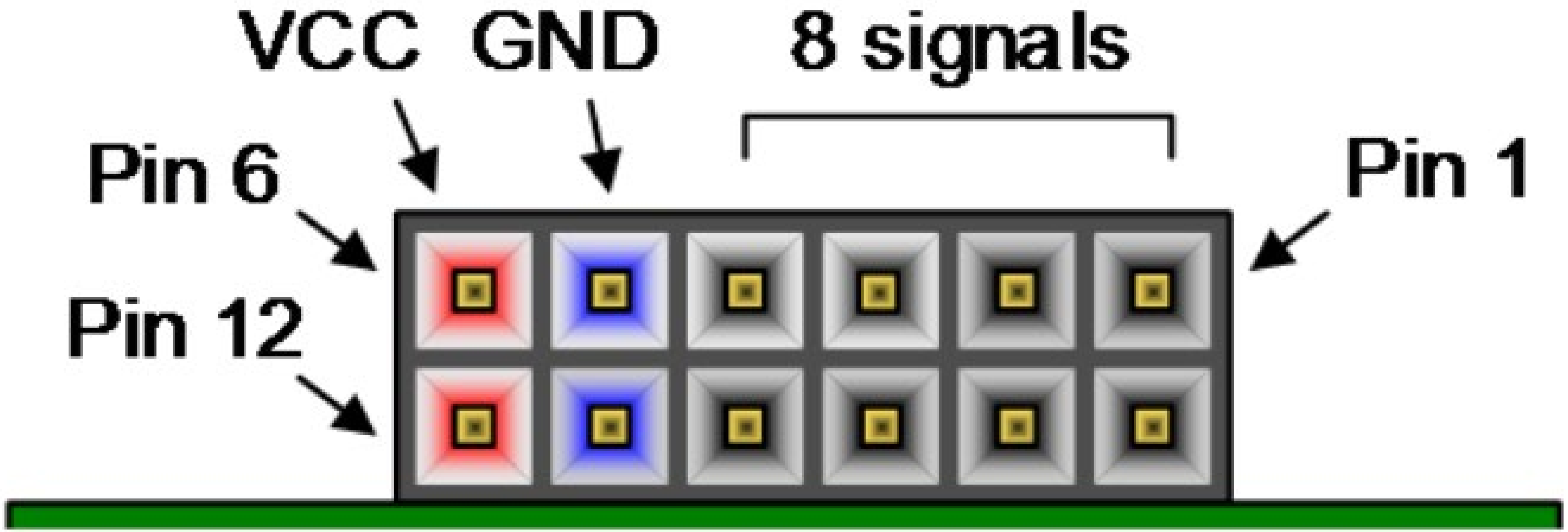
Transceiver Pins	
E	MGTAVCC_G#
V	MGTAVTT_G#
Λ	MGTVCCAUX_G#
<	MGTAVTTRCAL
G	MGTRREF
◆	MGTREFCLK1/0P
◆	MGTREFCLK1/0N
◇	MGTPRXP
◇	MGTPRXN
◇	MGTPTXP
◇	MGTPTXN

Dedicated Pins			
C	CCLK_0	S	VP_0
◨	CFGBVS_0	S	VN_0
D	DONE_0	S	VREFP_0
J	DXP_0	S	VREFN_0
L	DXN_0		
■	GNDADC_0		
Y	INIT_B_0		
0	M0_0		
1	M1_0		
2	M2_0		
P	PROGRAM_B_0		
K	TCK_0		
I	TDI_0		
O	TDO_0		
M	TMS_0		
◨	VCCADC_0		
⊞	VCCBATT_0		

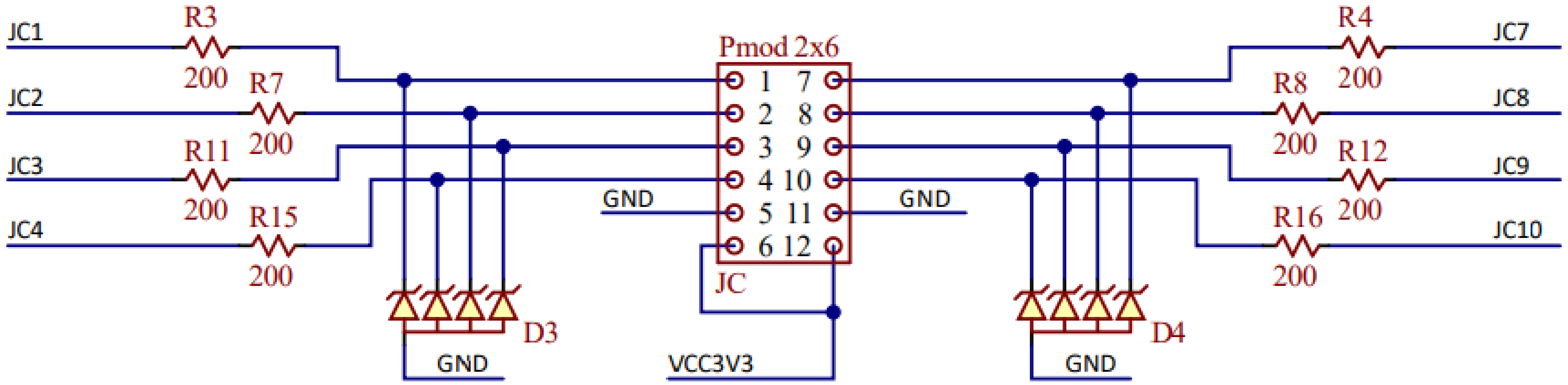
Other Pins	
▨	GND
◨	VCCAUX_IO_G#
◩	VCCAUX
▚	VCCINT
⧄	VCCO_#
◧	VCCBRAM
n	NC

Basys3 PMOD Connectors

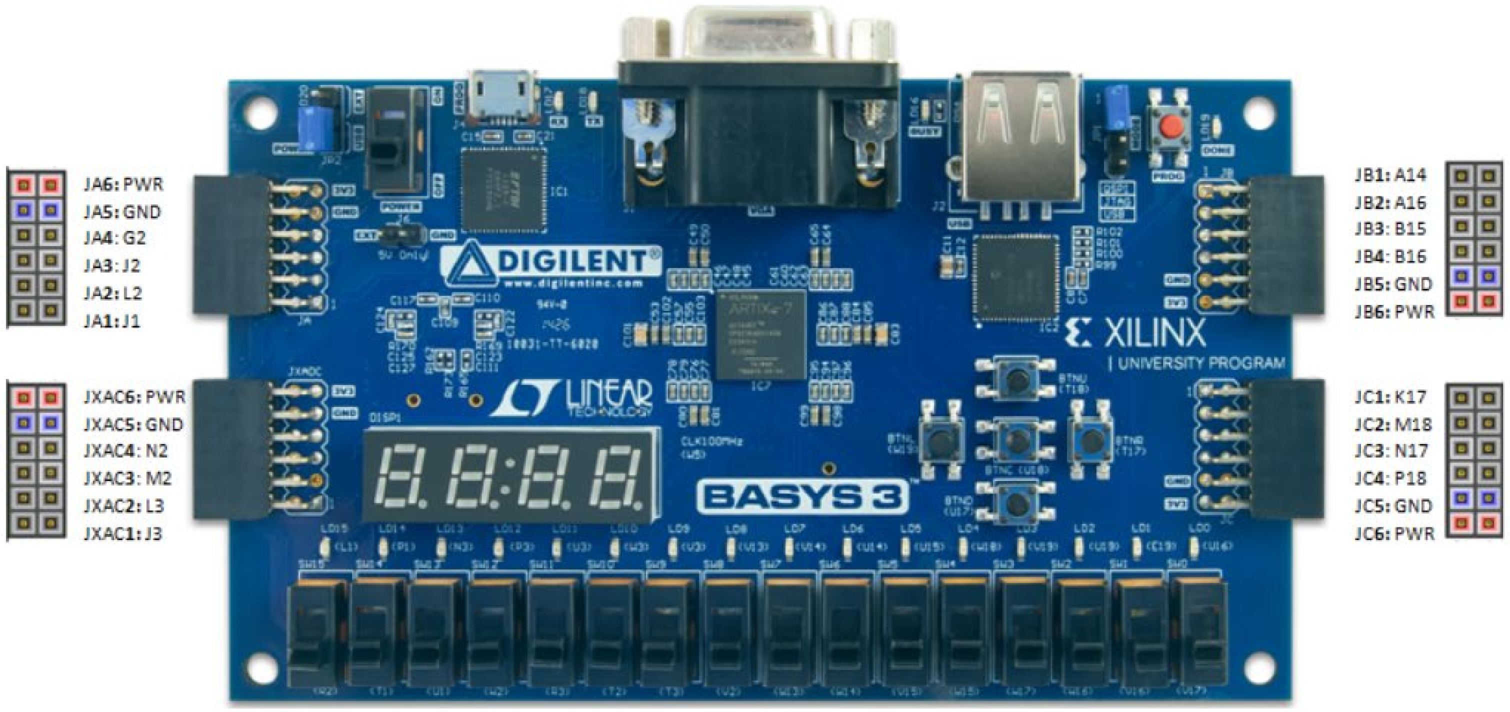

Pmod Connectors x4
DIGILENT
www.digilentinc.com
BASYS 3
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
ARTIX-7
JA
JXADC
JB
JC


VCC
GND
8 signals
Pin 6
Pin 12
Pin 1

Pmod JA	Pmod JB	Pmod JC	Pmod JXADC
JA1: J1	JB1: A14	JC1: K17	JXADC1: J3
JA2: L2	JB2: A16	JC2: M18	JXADC2: L3
JA3: J2	JB3: B15	JC3: N17	JXADC3: M2
JA4: G2	JB4: B16	JC4: P18	JXADC4: N2
JA7: H1	JB7: A15	JC7: L17	JXADC7: K3
JA8: K2	JB8: A17	JC8: M19	JXADC8: M3
JA9: H2	JB9: C15	JC9: P17	JXADC9: M1
JA10: G3	JB10: C16	JC10: R18	JXADC10: N1


JC1
R3
200
JC2
R7
200
JC3
R11
200
JC4
R15
200
Pmod 2x6
1 7
2 8
3 9
4 10
5 11
6 12
JC
GND
GND
D3
GND
VCC3V3
D4
GND
R4
200
JC7
R8
200
JC8
R12
200
JC9
R16
200
JC10

Basys3: Pmod Pin-Out Diagram



JA12 : PWR	JA6: PWR
JA11 : GND	JA5: GND
JA10: G3	JA4: G2
JA9: H2	JA3: J2
JA8 : K2	JA2: L2
JA7: H1	JA1: J1

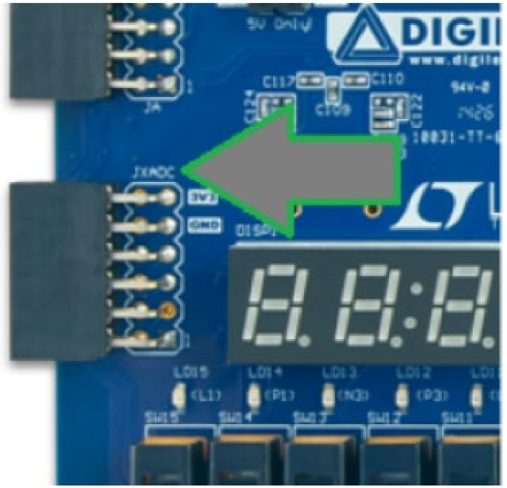
JXAC12 : PWR	JXAC6: PWR
JXAC11 : GND	JXAC5: GND
JXAC10: N1	JXAC4: N2
JXAC9: M1	JXAC3: M2
JXAC8 : M3	JXAC2: L3
JXAC7: K3	JXAC1: J3

JB1: A14	JB7: A15
JB2: A16	JB8: A17
JB3: B15	JB9: C15
JB4: B16	JB10: C16
JB5: GND	JB11: GND
JB6: PWR	JB12: PWR

JC1: K17	JC7: L17
JC2: M18	JC8: M19
JC3: N17	JC9: P17
JC4: P18	JC10: R18
JC5: GND	JC11: GND
JC6: PWR	JC12: PWR

Analog Inputs

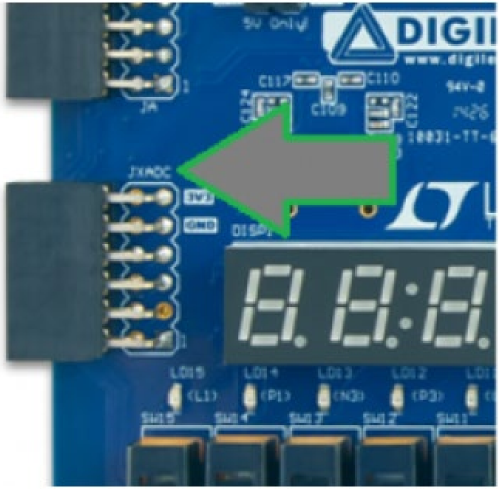
Dual Analog/Digital Pmod (JXADC)



Analog Inputs

1. Dual A2D Converters
2. 12-Bit
3. 1 MSPS
4. Built in Sensors
   - Temperature
   - Power Rails

Dual Analog/Digital Pmod (JXADC)



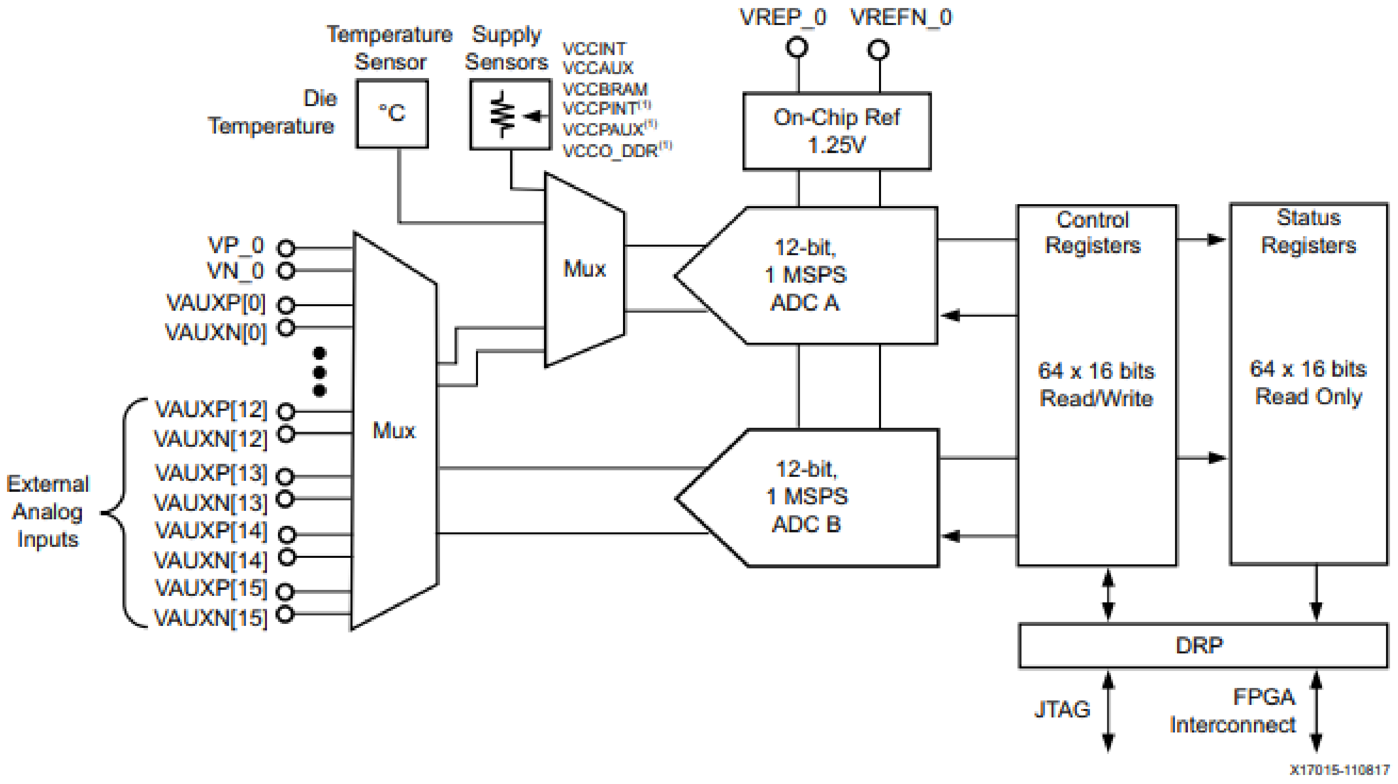
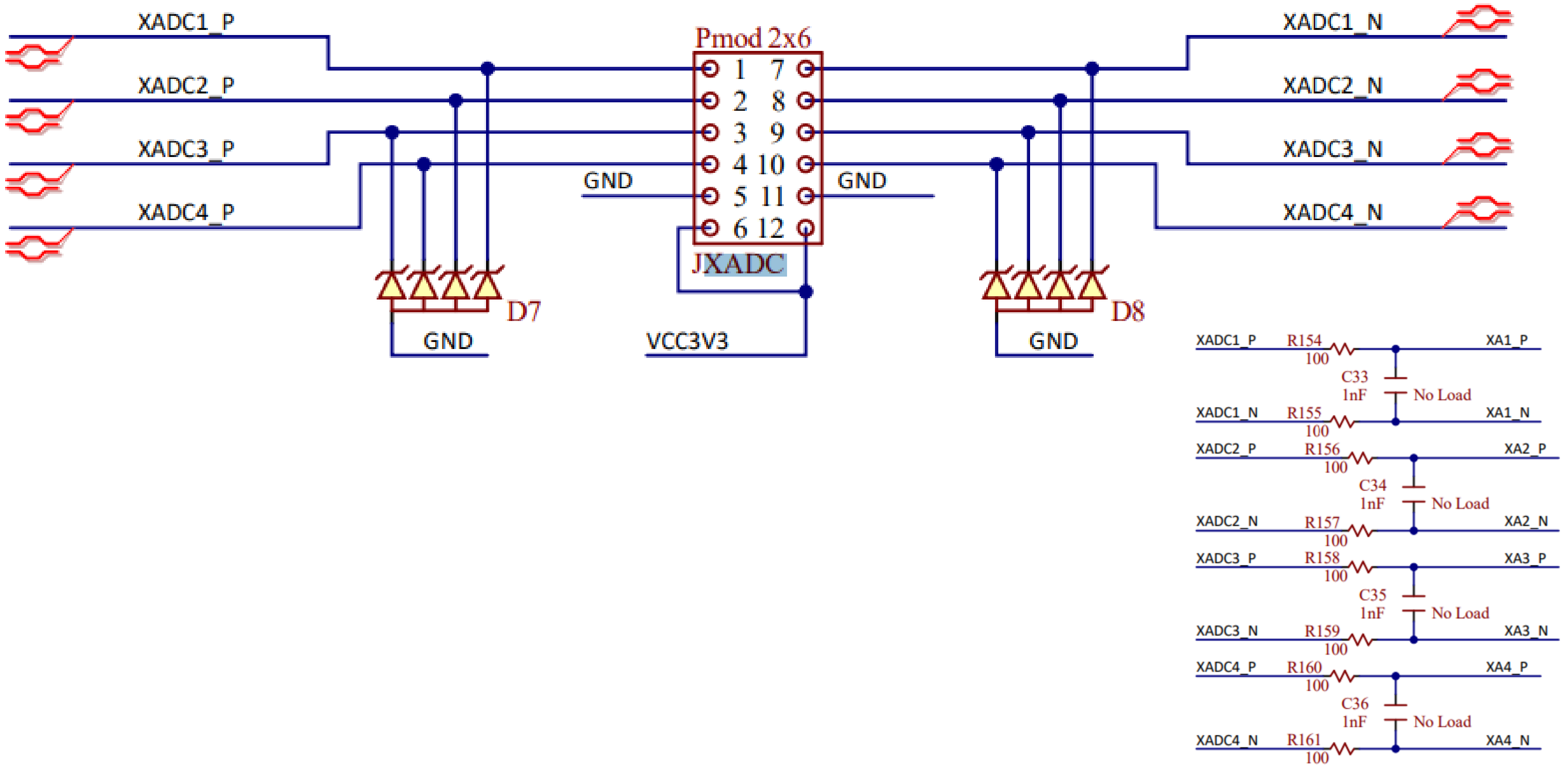


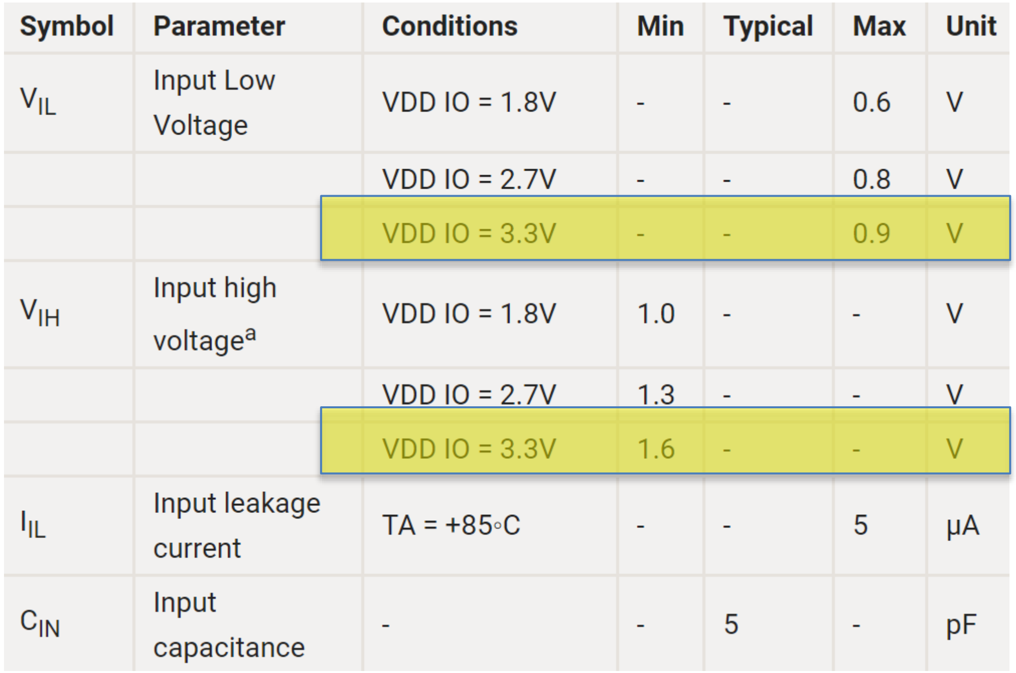
Figure 1-1: **XADC Block Diagram**

https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf


XADC1_P
XADC2_P
XADC3_P
XADC4_P
Pmod 2x6
1 7
2 8
3 9
4 10
5 11
6 12
JXADC
GND
GND
D7
D8
GND
GND
VCC3V3
XADC1_N
XADC2_N
XADC3_N
XADC4_N
XADC1_P R154 100 XA1_P
C33 1nF No Load
XADC1_N R155 100 XA1_N
XADC2_P R156 100 XA2_P
C34 1nF No Load
XADC2_N R157 100 XA2_N
XADC3_P R158 100 XA3_P
C35 1nF No Load
XADC3_N R159 100 XA3_N
XADC4_P R160 100 XA4_P
C36 1nF No Load
XADC4_N R161 100 XA4_N

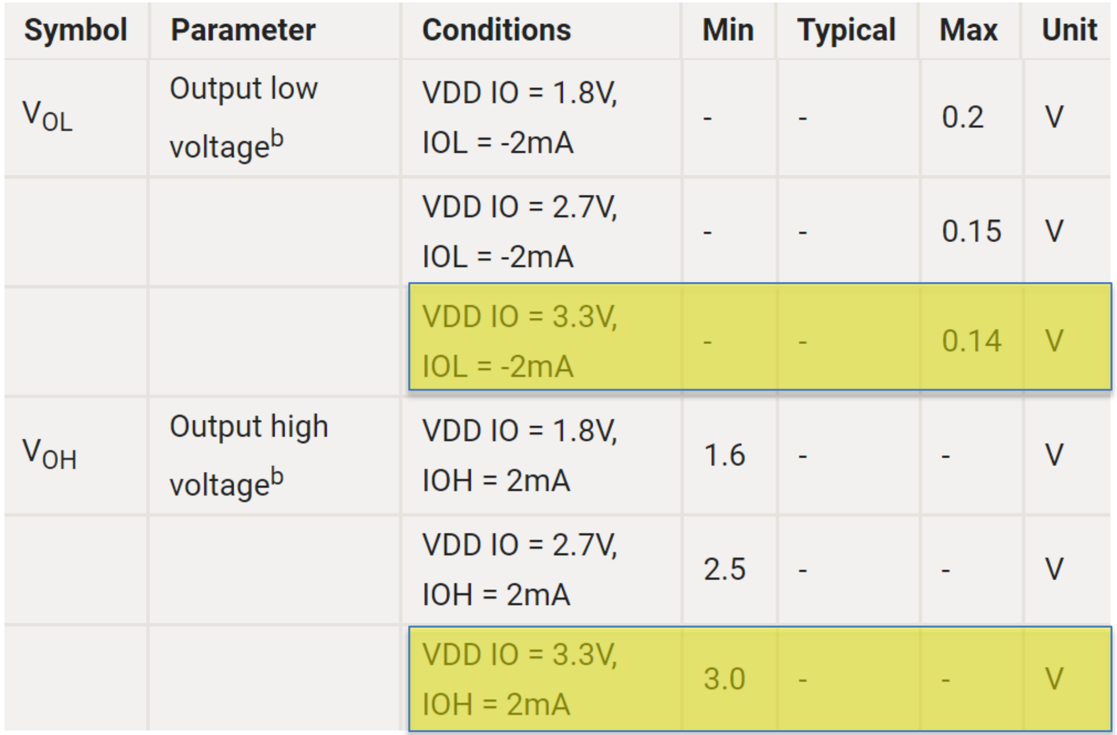
Pin Voltages

Raspberry Pin Input Pins



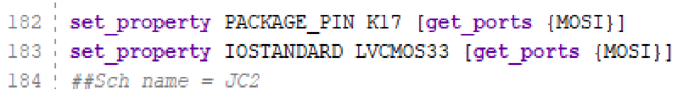
Symbol	Parameter	Conditions	Min	Typical	Max	Unit
V_{IL}	Input Low Voltage	VDD IO = 1.8V	-	-	0.6	V
		VDD IO = 2.7V	-	-	0.8	V
		VDD IO = 3.3V	-	-	0.9	V
V_{IH}	Input high voltage[a]	VDD IO = 1.8V	1.0	-	-	V
		VDD IO = 2.7V	1.3	-	-	V
		VDD IO = 3.3V	1.6	-	-	V
I_{IL}	Input leakage current	TA = +85◦C	-	-	5	μA
C_{IN}	Input capacitance	-	-	5	-	pF

Raspberry Pi Output Pins



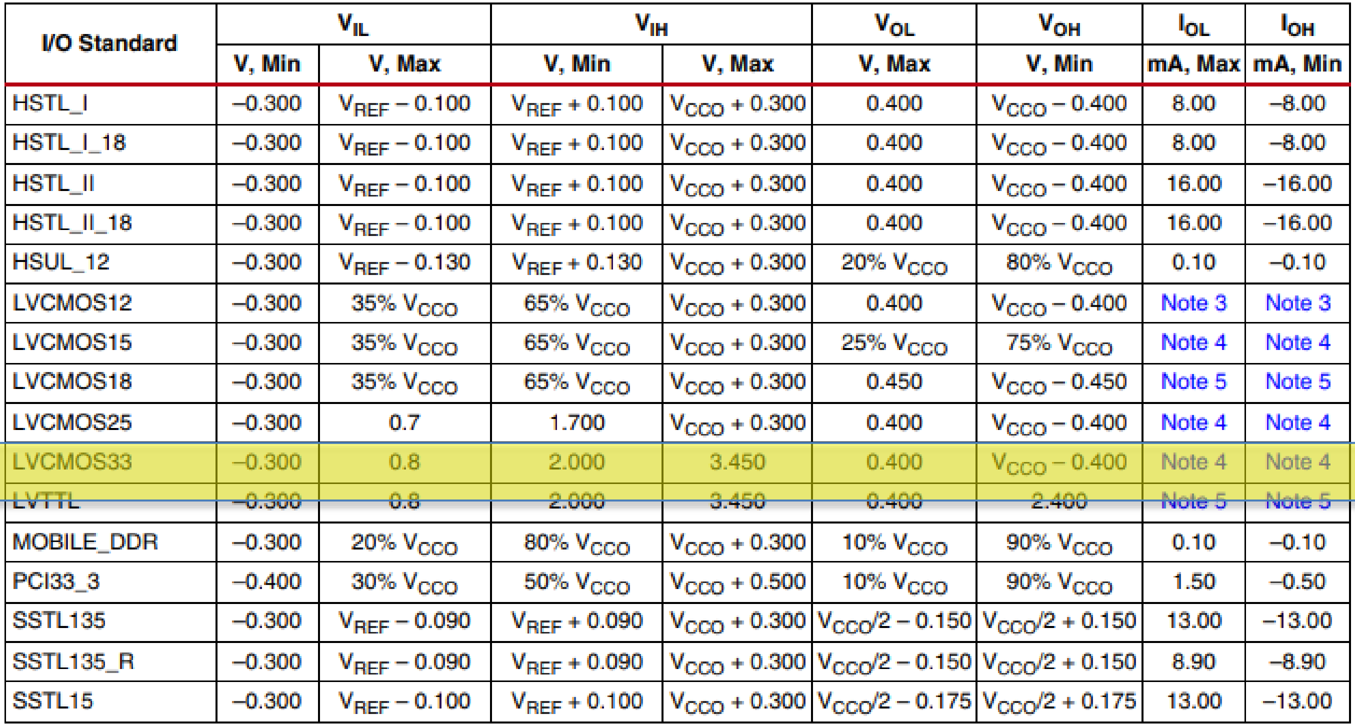
Symbol	Parameter	Conditions	Min	Typical	Max	Unit
V_{OL}	Output low voltage[b]	VDD IO = 1.8V, IOL = -2mA	-	-	0.2	V
		VDD IO = 2.7V, IOL = -2mA	-	-	0.15	V
		VDD IO = 3.3V, IOL = -2mA	-	-	0.14	V
V_{OH}	Output high voltage[b]	VDD IO = 1.8V, IOH = 2mA	1.6	-	-	V
		VDD IO = 2.7V, IOH = 2mA	2.5	-	-	V
		VDD IO = 3.3V, IOH = 2mA	3.0	-	-	V

Xilinx Artix XC7A35T FPGA



```
set_property PACKAGE_PIN K17 [get_ports {MOSI}]
set_property IOSTANDARD LVCMOS33 [get_ports {MOSI}]
##Sch name = JC2
```

Table 8: **SelectIO DC Input and Output Levels**(1)(2)



I/O Standard	V_{IL}		V_{IH}		V_{OL}	V_{OH}	I_{OL}	I_{OH}
	V, Min	V, Max	V, Min	V, Max	V, Max	V, Min	mA, Max	mA, Min
HSTL_I	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	8.00	–8.00
HSTL_I_18	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	8.00	–8.00
HSTL_II	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	16.00	–16.00
HSTL_II_18	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	16.00	–16.00
HSUL_12	–0.300	V_{REF} – 0.130	V_{REF} + 0.130	V_{CCO} + 0.300	20% V_{CCO}	80% V_{CCO}	0.10	–0.10
LVCMOS12	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	Note 3	Note 3
LVCMOS15	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	25% V_{CCO}	75% V_{CCO}	Note 4	Note 4
LVCMOS18	–0.300	35% V_{CCO}	65% V_{CCO}	V_{CCO} + 0.300	0.450	V_{CCO} – 0.450	Note 5	Note 5
LVCMOS25	–0.300	0.7	1.700	V_{CCO} + 0.300	0.400	V_{CCO} – 0.400	Note 4	Note 4
LVCMOS33	–0.300	0.8	2.000	3.450	0.400	V_{CCO} – 0.400	Note 4	Note 4
LVTTL	–0.300	0.8	2.000	3.450	0.400	2.400	Note 5	Note 5
MOBILE_DDR	–0.300	20% V_{CCO}	80% V_{CCO}	V_{CCO} + 0.300	10% V_{CCO}	90% V_{CCO}	0.10	–0.10
PCI33_3	–0.400	30% V_{CCO}	50% V_{CCO}	V_{CCO} + 0.500	10% V_{CCO}	90% V_{CCO}	1.50	–0.50
SSTL135	–0.300	V_{REF} – 0.090	V_{REF} + 0.090	V_{CCO} + 0.300	V_{CCO}/2 – 0.150	V_{CCO}/2 + 0.150	13.00	–13.00
SSTL135_R	–0.300	V_{REF} – 0.090	V_{REF} + 0.090	V_{CCO} + 0.300	V_{CCO}/2 – 0.150	V_{CCO}/2 + 0.150	8.90	–8.90
SSTL15	–0.300	V_{REF} – 0.100	V_{REF} + 0.100	V_{CCO} + 0.300	V_{CCO}/2 – 0.175	V_{CCO}/2 + 0.175	13.00	–13.00

https://www.xilinx.com/support/documentation/data_sheets/ds181_Artix_7_Data_Sheet.pdf

Shift Registers

4 Bit Shift Register

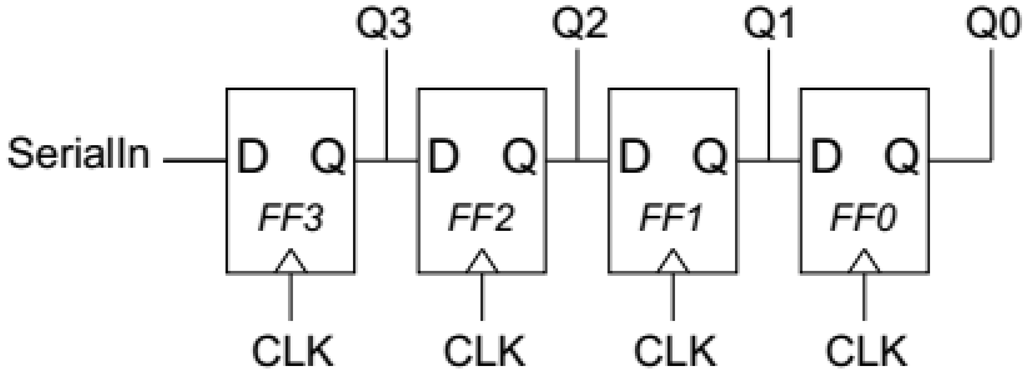
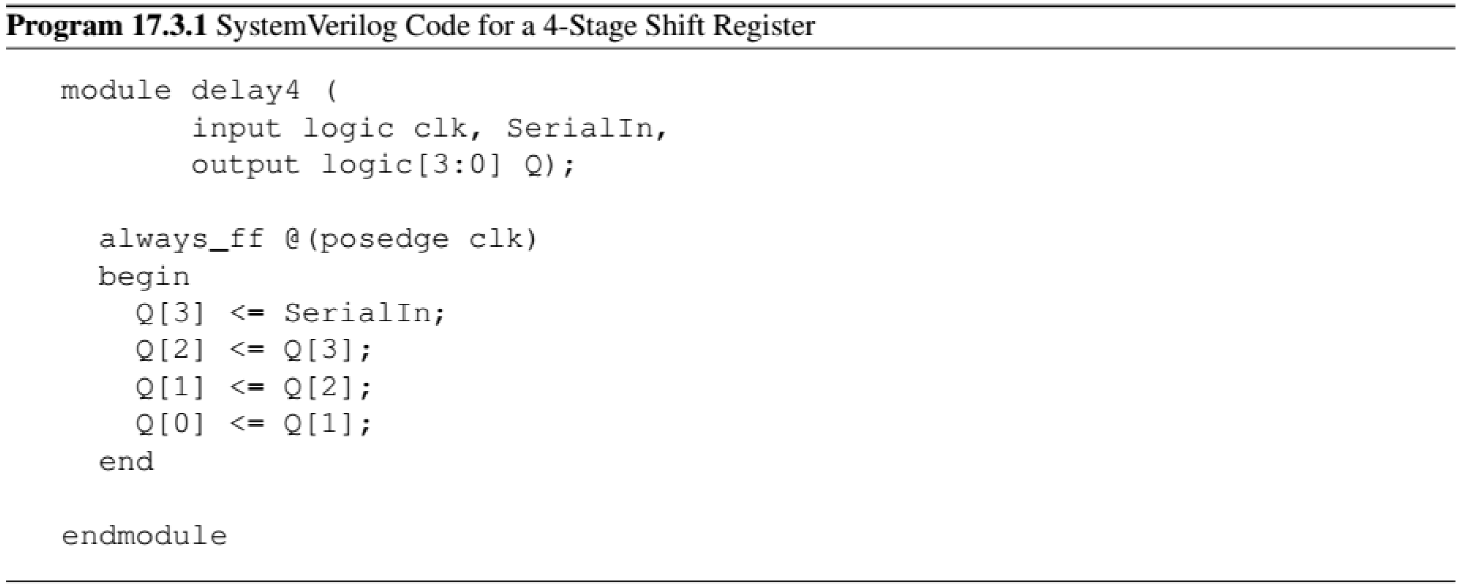


Figure 16.9: A Shift Register Made from Four D Flip Flops

Source: Nelson (2019) Designing Digital Systems with SystemVerilog (v2.0)

4 Bit Shift Register (Behavioral SystemVerilog)

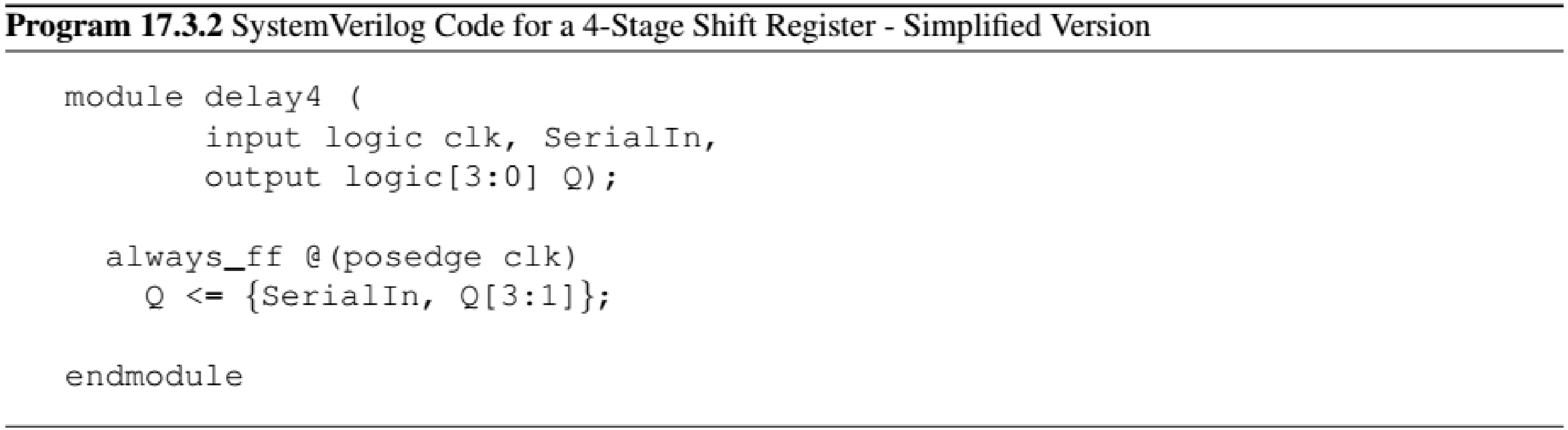


Program 17.3.1 SystemVerilog Code for a 4-Stage Shift Register

```
module delay4 (
       input logic clk, SerialIn,
       output logic[3:0] Q);

  always_ff @(posedge clk)
  begin
    Q[3] <= SerialIn;
    Q[2] <= Q[3];
    Q[1] <= Q[2];
    Q[0] <= Q[1];
  end

endmodule
```

[Source: Nelson (2019) Designing Digital Systems with SystemVerilog (v2.0)]

Refinement …



Program 17.3.2 SystemVerilog Code for a 4-Stage Shift Register - Simplified Version

```
module delay4 (
       input logic clk, SerialIn,
       output logic[3:0] Q);

  always_ff @(posedge clk)
    Q <= {SerialIn, Q[3:1]};

endmodule
```

Source: Nelson (2019) Designing Digital Systems with SystemVerilog (v2.0)

What is the effect of this reordering?

```
module delay4 (
       input logic clk, SerialIn,
       output logic[3:0] Q);

  always_ff @(posedge clk)
  begin


    Q[2] <= Q[3];
    Q[0] <= Q[1];
    Q[3] <= SerialIn;
    Q[1] <= Q[2];
  end

endmodule
```

Source: Nelson (2019) Designing Digital Systems with SystemVerilog (v2.0)

Asynchronous Input

Consider the following general architecture …

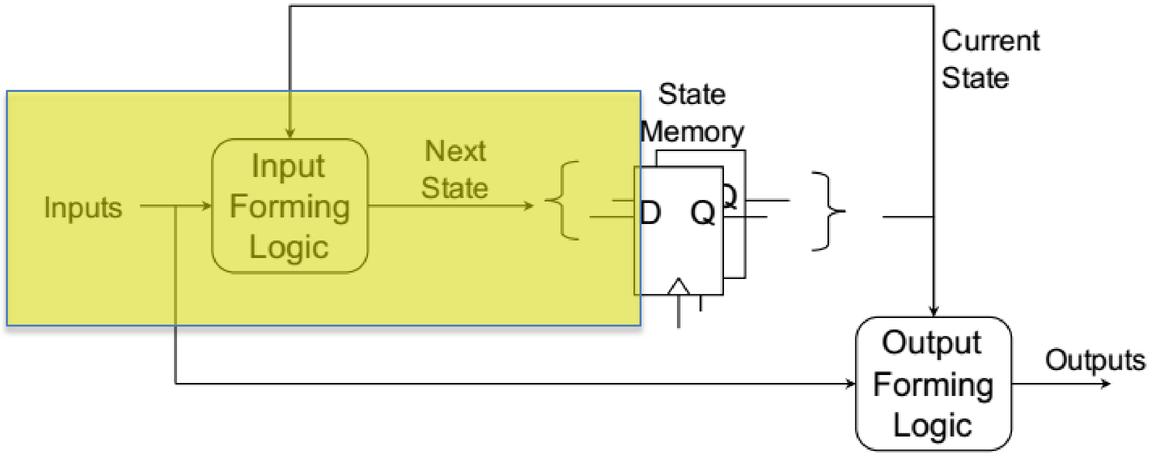
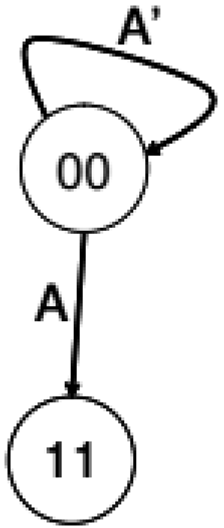


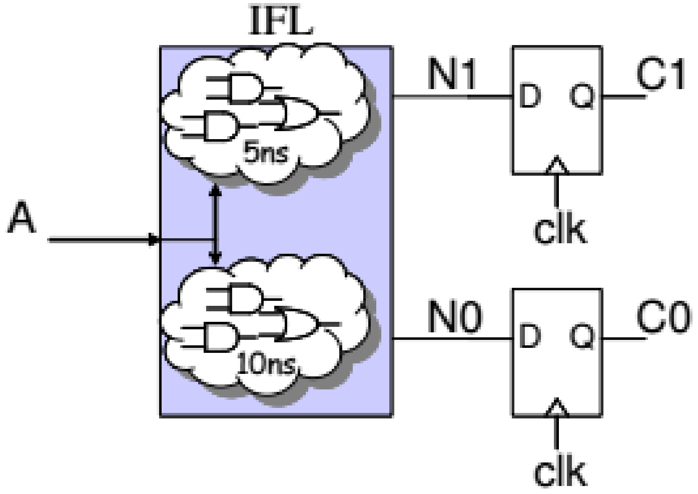
Figure 16.12: Register Transfer Level (RTL) Structure

Asynchronous Input

1. Simplifying Design Assumption: combinational logic stabilizes before the sequential circuitry uses the resulting values.
2. Consider signals from a different clock domain
   - Raspberry Pi…

Consider the following state machine.





Assume A arrives from a different clock domain.
Arrival of A can be any time relative to clk.

Consider the following timing …

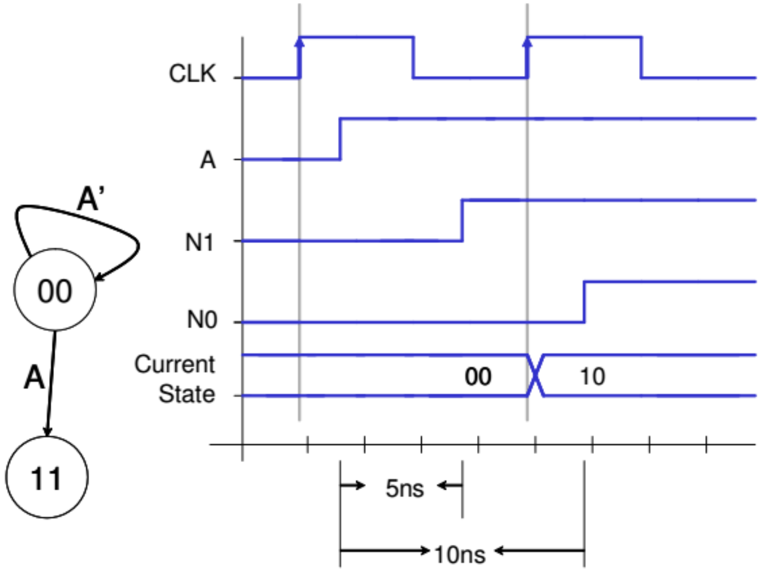


Figure 24.2: Timing Problem of Figure 24.1

Solution: Add a flip flop …

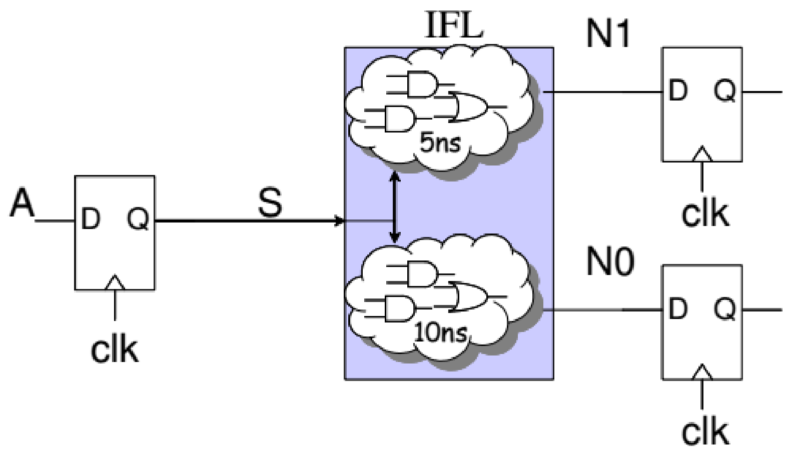


Figure 24.3: Synchronizing an Asynchronous Input

No free lunch … A is now stable but delayed by 1 clock.

Asynchronous Output

Consider the following general architecture …

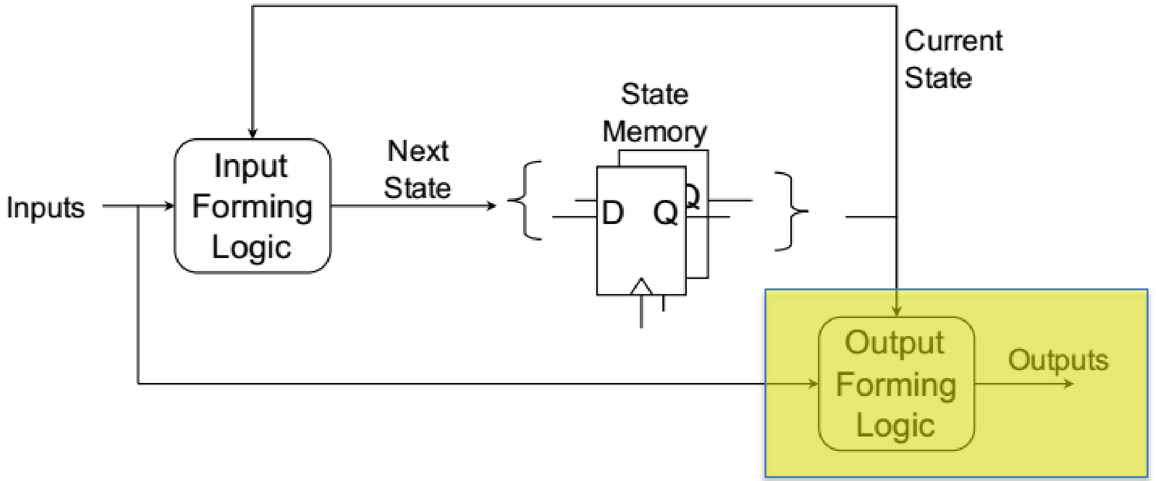
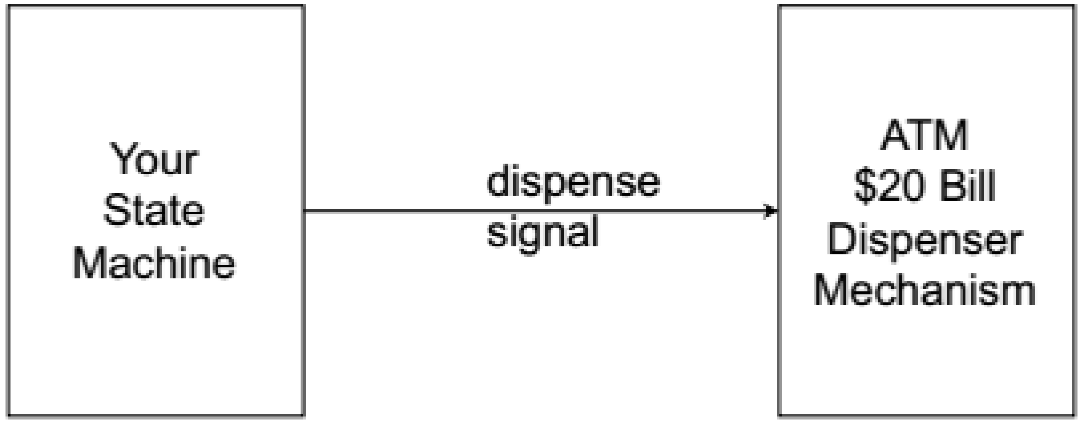


Figure 16.12: Register Transfer Level (RTL) Structure

Consider the following ATM signal



If dispense signal were going to another module within the same clock domain, no issue. What if this were a different clock domain?

Synchronize combinational logic before output

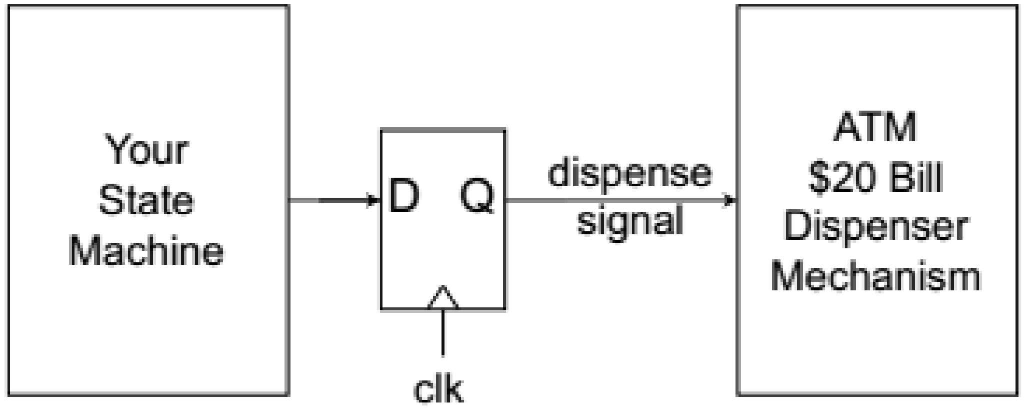


Figure 24.5: Generating a Glitch-Free Output