Department of Electronics & Communication Engg.

Course : Digital System Design using Verilog. **Sem.: 6th (2017-18)**

Course Coordinator:

Prof. D. M. Kumbhar

Digital System Design Using Verilog

Module 4

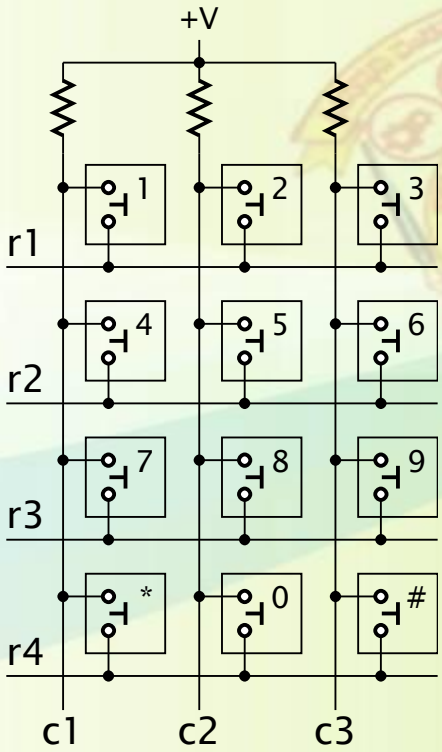
I/O Interfacing

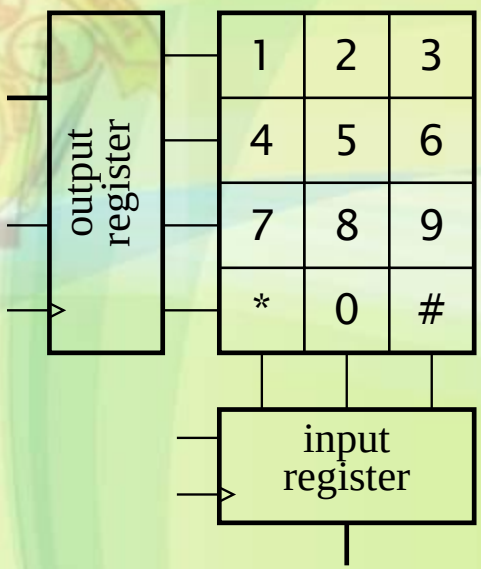
I/O Devices and Transducers

- Transducers convert between real-world effects and digital representation
  - Input transducers: sensors
    - May require analog-to-digital converter (ADC)
  - Output transducers: actuators
    - May require digital-to-analog converter (DAC)
- Human-interface devices
  - Buttons, switches, knobs, keypads, mouse
  - Indicators, displays, speakers

Kaypads & Keyboards

- Recall switches and debouncing
- Keypad: array of push-button switches



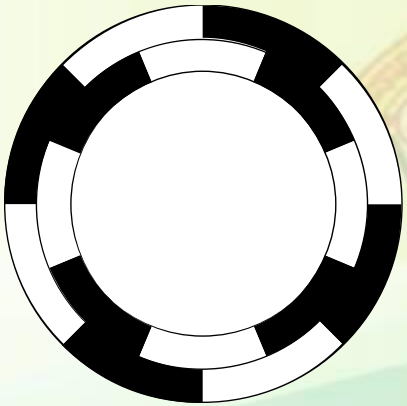


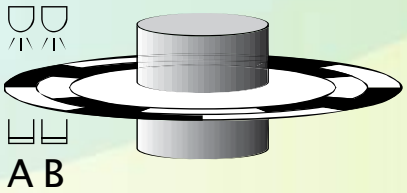
Knobs & Position Encoders

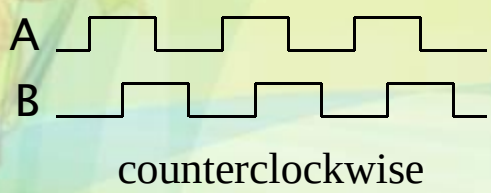
- In analog circuits, use a variable resistor
- In digital circuits, could use pushbuttons
  - E.g., volume up/down
  - Not as easy to use as knobs or sliders
- Can use a position encoder attached to a knob
  - Recall Gray code encoder

Incremental Encoder

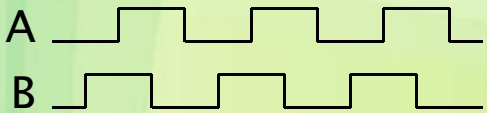
- If absolute position is not important, incremental encoder is simpler







counterclockwise



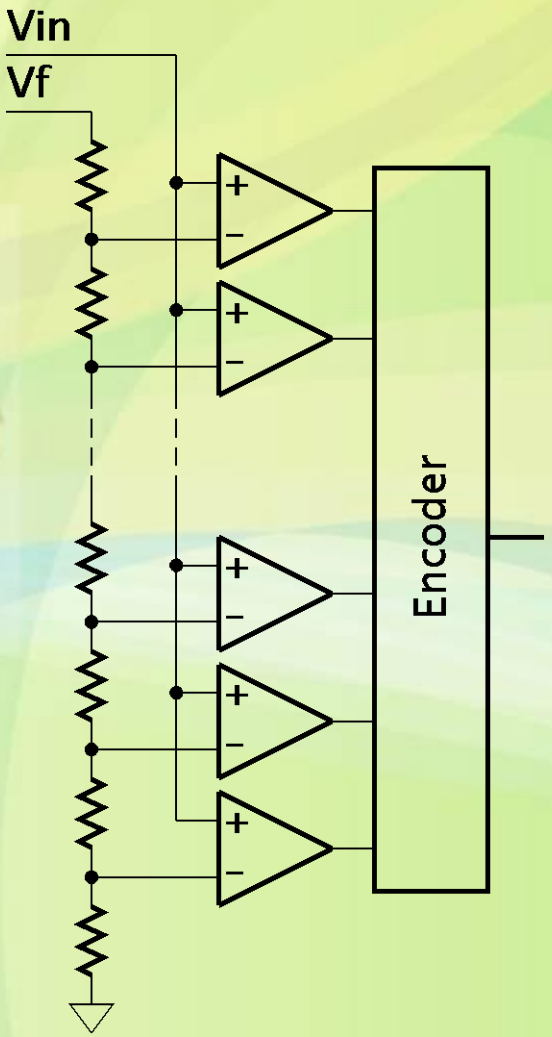
clockwise

Analog Inputs

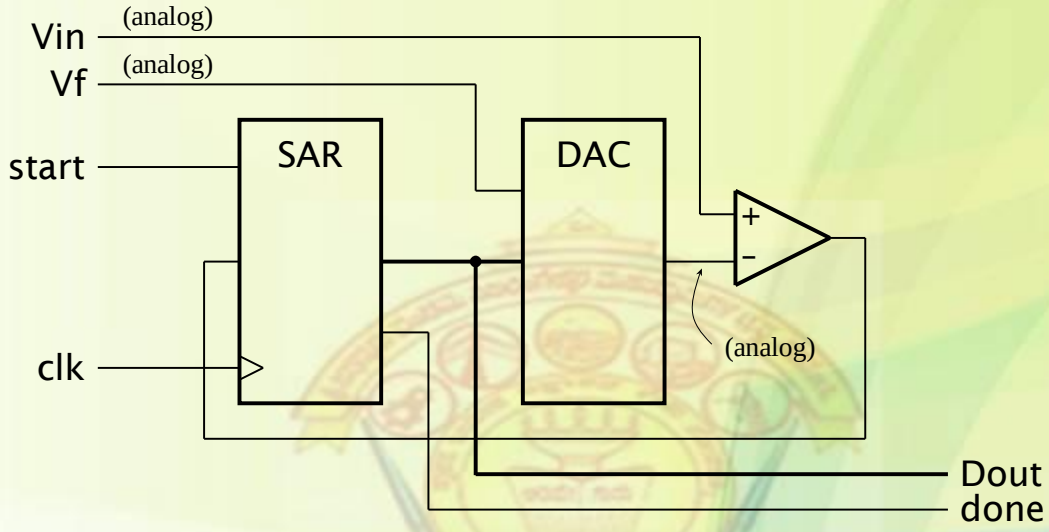
- Physical effect produces an analog voltage or current
- Microphone
  - In phones, cameras, voice recorders, …
- Accelerometer
  - In airbag controllers
- Fluid-flow sensors
  - In industrial machines, coffee machines, …
- Gas detectors
  - In safety equipment

Analog-to-Digital Converters

- Basic element: analog comparator
- Flash ADC
  - For n o/p 2^n-1 comparators
  - Simple, fast, but uses many comparators
- Resolution
  - Number of output bits



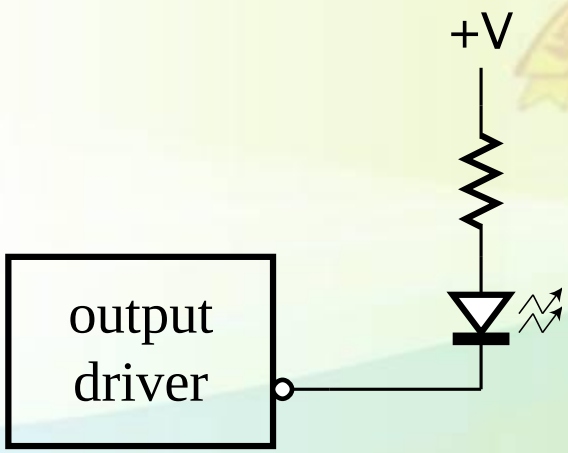
Successive Approximation ADC



- Initial approximation: 01111111
  - Comparator output gives d_7
    - 1 if Vin is higher than 01111111, 0 otherwise
- Next approximation: $d_7$0111111
  - Comparator output gives d_6
- Next approximation: $d_7d_6$011111, etc

LED Indicators

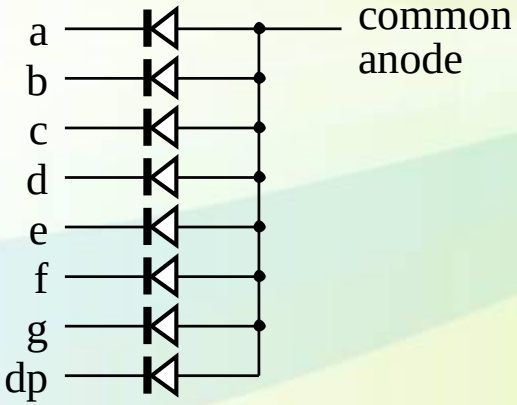
- Single LED shows 1-bit state
  - On/off, busy/ready, error/ok, …

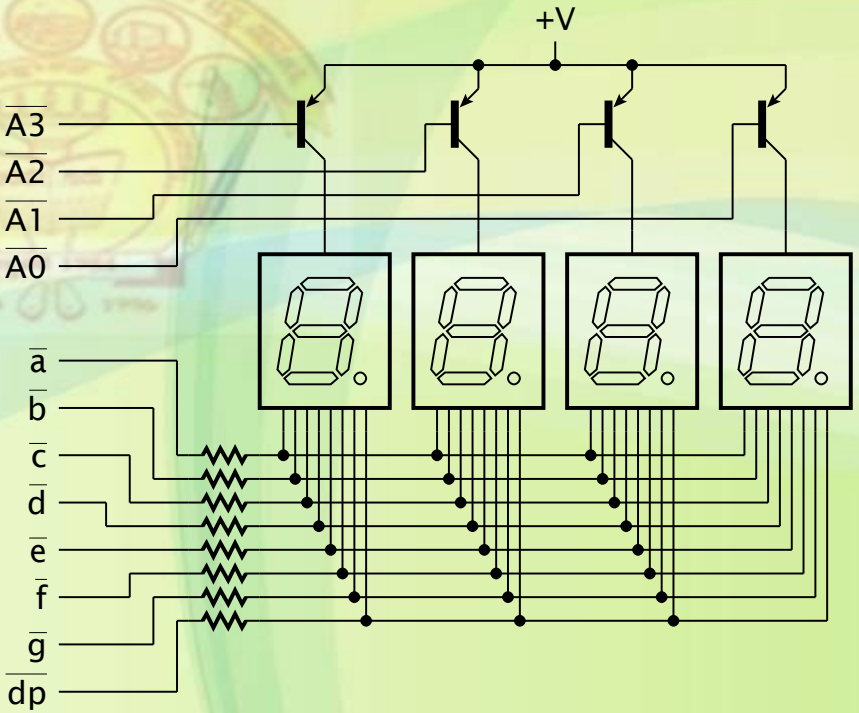


- Brightness depends on current
  - Determined by resistor
  - $I = (+V - V_{LED} - V_{OL}) / R$

7-Segment LED Displays

- Each digit has common anodes or common cathodes
  - Scan: turn on one digit at a time





Example: Multiplexed Display

- Four BDC inputs, 10MHz clock
  - Turn on decimal point of leftmost digit only
  - 50Hz scan cycle (200Hz scan clock)

```
module display_mux ( output reg [3:0] anode_n,
                     output     [7:0] segment_n,
                     input      [3:0] bcd0, bcd1, bcd2, bcd3,
                     input            clk, reset );

  parameter clk_freq      = 10000000;
  parameter scan_clk_freq = 200;
  parameter clk_divisor   = clk_freq / scan_clk_freq;

  reg       scan_clk;
  reg [1:0] digit_sel;
  reg [3:0] bcd;
  reg [7:0] segment;

  integer count;
```

Example: Multiplexed Display

```
// Divide master clock to get scan clock
always @(posedge clk)
  if (reset) begin
    count = 0;
    scan_clk <= 1'b0;
  end
  else if (count == clk_divisor - 1) begin
    count = 0;
    scan_clk <= 1'b1;
  end
  else begin
    count = count + 1;
    scan_clk <= 1'b0;
  end

// increment digit counter once per scan clock cycle
always @(posedge clk)
  if      (reset)    digit_sel <= 2'b00;
  else if (scan_clk) digit_sel <= digit_sel + 1;
```

Example: Multiplexed Display

```
// multiplexer to select a BCD digit
always @*
  case (digit_sel)
    2'b00: bcd = bcd0;
    2'b01: bcd = bcd1;
    2'b10: bcd = bcd2;
    2'b11: bcd = bcd3;
  endcase

// activate selected digit's anode
always @*
  case (digit_sel)
    2'b00: anode_n = 4'b1110;
    2'b01: anode_n = 4'b1101;
    2'b10: anode_n = 4'b1011;
    2'b11: anode_n = 4'b0111;
  endcase
```

Example: Multiplexed Display

```
// 7-segment decoder for selected digit
always @*
  case (bcd)
    4'b0000: segment[6:0] = 7'b0111111;  // 0
    4'b0001: segment[6:0] = 7'b0000110;  // 1
    4'b0010: segment[6:0] = 7'b1011011;  // 2
    4'b0011: segment[6:0] = 7'b1001111;  // 3
    4'b0100: segment[6:0] = 7'b1100110;  // 4
    4'b0101: segment[6:0] = 7'b1101101;  // 5
    4'b0110: segment[6:0] = 7'b1111101;  // 6
    4'b0111: segment[6:0] = 7'b0000111;  // 7
    4'b1000: segment[6:0] = 7'b1111111;  // 8
    4'b1001: segment[6:0] = 7'b1101111;  // 9
    default: segment[6:0] = 7'b1000000;  // "-"
  endcase
```

Example: Multiplexed Display

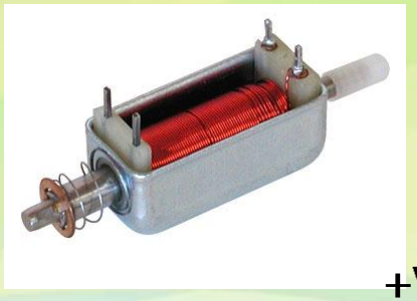
```
  // decimal point is only active for digit 3
  always @* segment[7] = digit_sel == 2'b11;

  // segment outputs are negative logic
  assign segment_n = ~segment;

endmodule
```

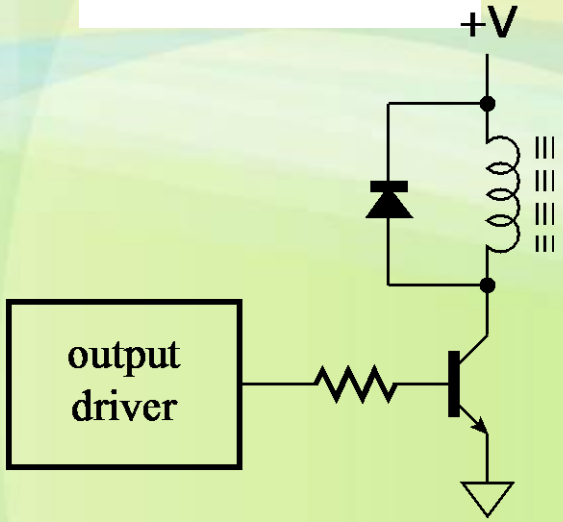
Liquid Crystal Displays (LCDs)

- Advantages
  - Low power
  - Readable in bright ambient light conditions
  - Custom segment shapes
- Disadvantages
  - Require backlight for dark conditions
  - Not as robust as LEDs
- LCD panels
  - Rectangular array of pixels
  - Can be used for alphanumeric/graphical display
  - Controlled by a small microcontroller

Actuators & Valves

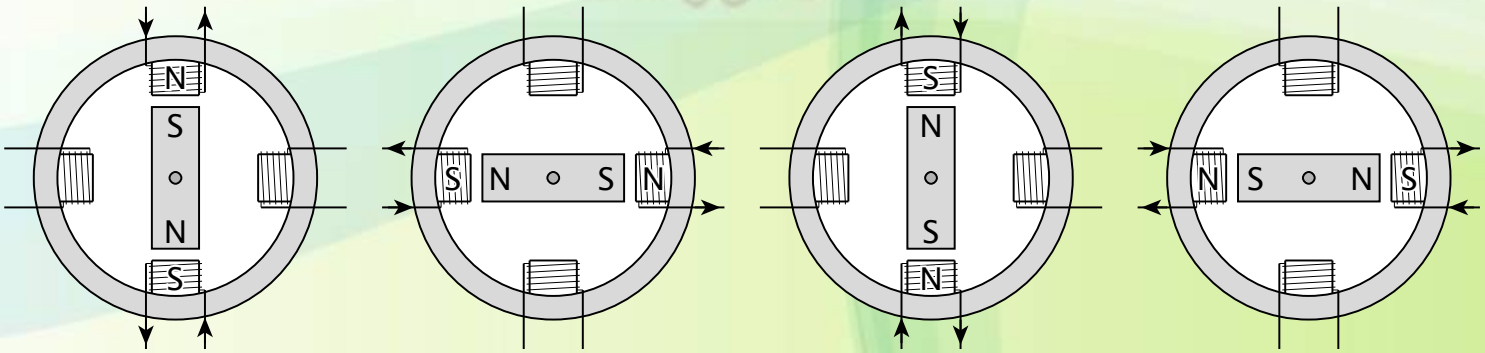
- Actuators cause a mechanical effect
- Solenoid: current in coil moves armature
  - Can attach rods, levers, etc to translate the movement
- Solenoid valve
  - Armature controls fluid or gas valve
- Relay
  - Armature controls electrical contacts





Motors

- Can provide angular position or speed
  - Use gears, screws, etc to convert to linear position or speed
- Stepper motors
  - Rotate in discrete steps

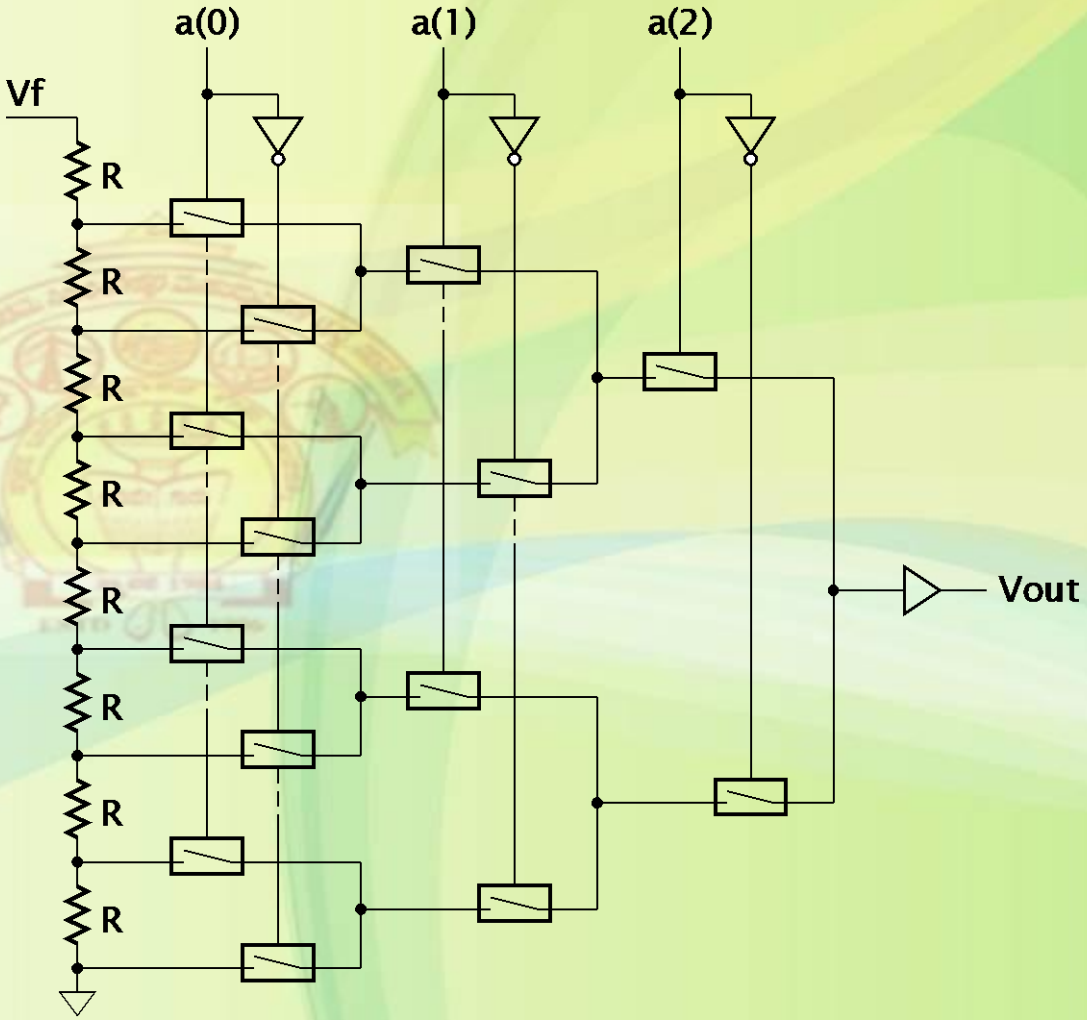


Motors

- Servo-motors
  - DC motor, speed controlled by varying the drive voltage
  - Use feedback to control the speed or to drive to a desired position
  - Requires a position sensor or tachometer
- Servo-controller
  - A digital circuit or an embedded processor
  - Compensates for non-ideal mechanical effects

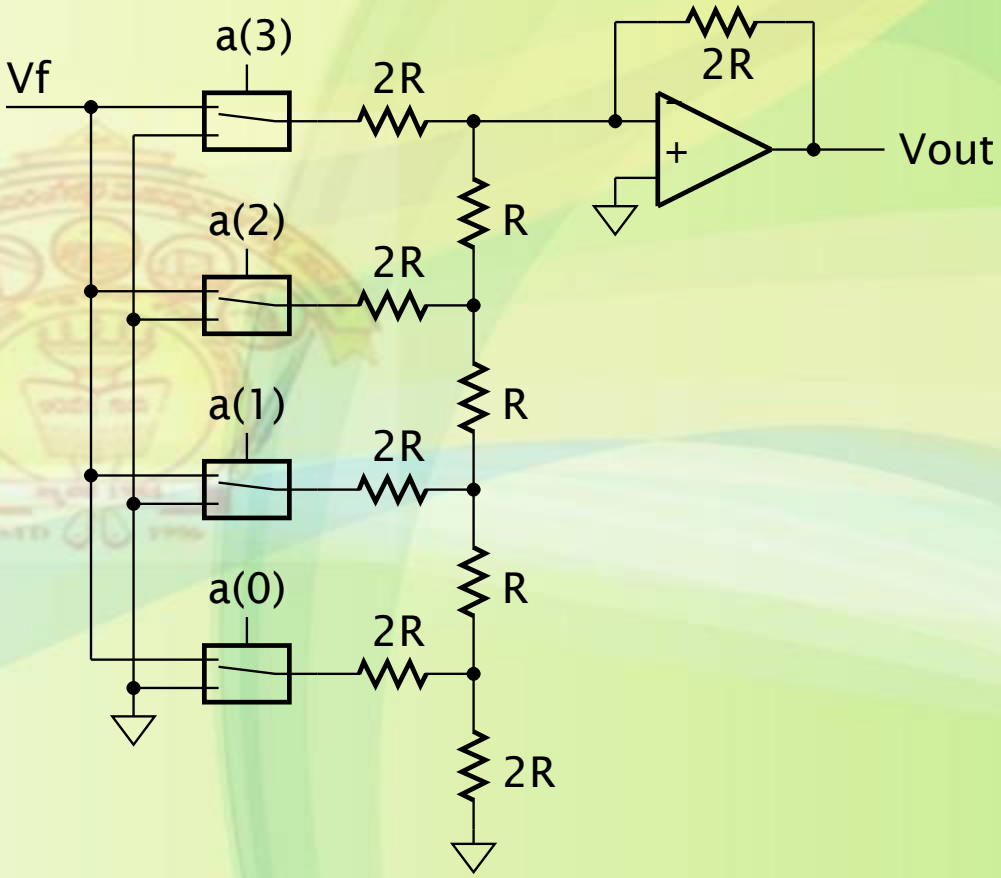
Digital-to-Analog Converters

- R-string DAC
  - Voltage divider and analog multiplexer
  - Requires 2^n precision resistors



Digital-to-Analog Converters

- R-2R ladder DAC
  - Sums binary-weighted currents
  - Requires 2*n* matched resistors



I/O Controllers

- An embedded processor needs to access input/output data
- I/O controller
  - Circuit that connects I/O device to a processor
    - Includes control circuits
  - Input registers: for reading data
  - Output registers: for writing data
- I/O ports
  - Registers accessible to embedded software

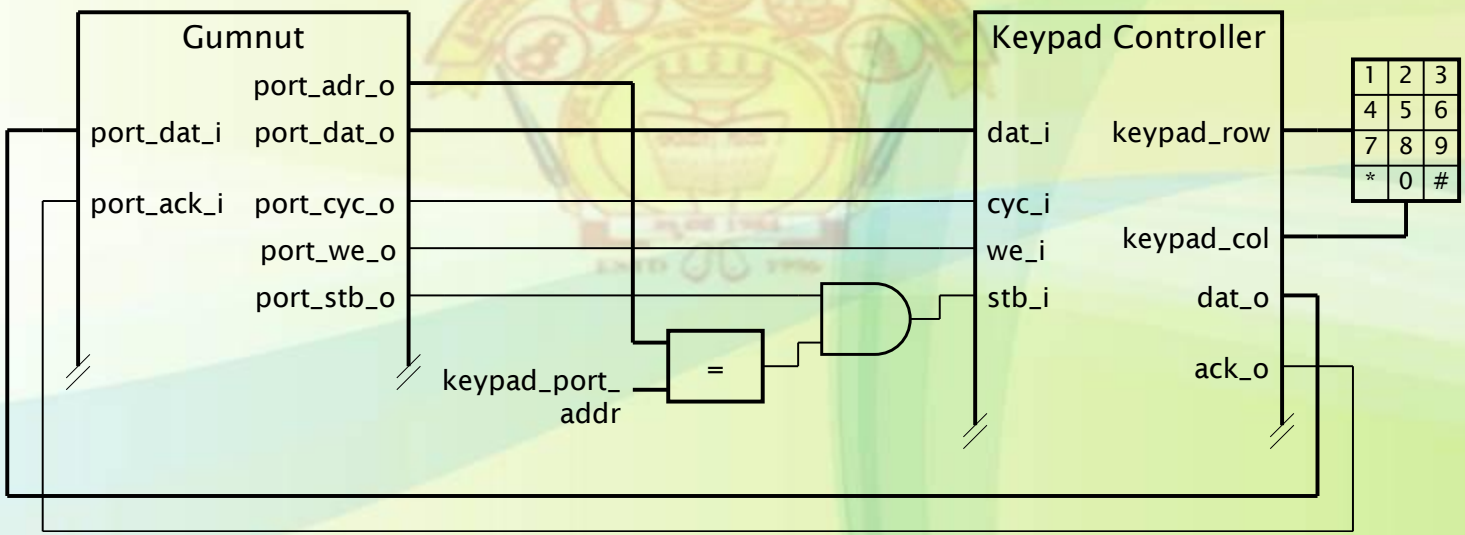
Simple I/O Controller

- Just contains input and/or output registers
  - Select among them using a port address

```
module gumnut ( input         clk_i,
                input         rst_i,
                ...
                output        port_cyc_o,
                output        port_stb_o,
                output        port_we_o,
                input         port_ack_i,
                output [7:0]  port_adr_o,
                output [7:0]  port_dat_o,
                input  [7:0]  port_dat_i,
                ... );
endmodule
```

Example: Keypad Controller

- Output register for row drivers
- Input register for column sensing



Example: Keypad Controller

```
module keypad_controller ( input             clk_i,
                           input             cyc_i,
                           input             stb_i,
                           input             we_i,
                           output            ack_o,
                           input      [7:0]  dat_i,
                           output reg [7:0]  dat_o,
                           output reg [3:0]  keypad_row,
                           input      [2:0]  keypad_col );

  reg [2:0] col_synch;

  always @(posedge clk_i)  // Row register
    if (cyc_i && stb_i && we_i) keypad_row <= dat_i[3:0];

  always @(posedge clk_i) begin  // Column synchronizer
    dat_o     <= {5'b0, col_synch};
    col_synch <= keypad_col;
  end

  assign ack_o = cyc_i && stb_i;

endmodule
```

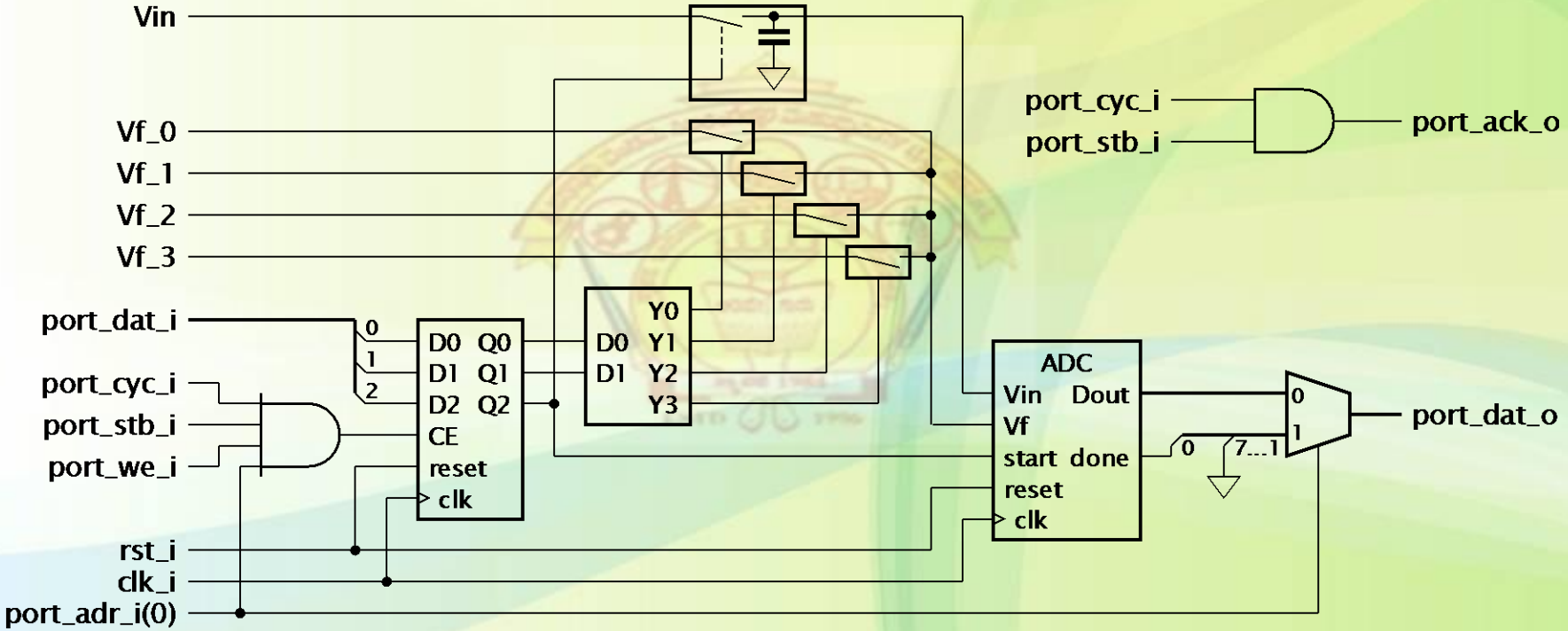
Control/Status Registers

- Control register
  - Contains bits that govern operation of the I/O device
  - Written by processor
- Status register
  - Contains bits that reflect status of device
  - Read by processor
- Either or both may be needed in an input or output controller

Example: ADC Controller

- Successive approximation ADC
  - 1 × analog input with sample/hold
  - 4 × analog reference voltages
- Control register
  - Selects reference voltage
  - Hold input voltage & start ADC
- Status register
  - Is conversion done?
- Input data register
  - Converted data

Example: ADC Controller



Autonomous I/O Controllers

- Independently sequence operation of a device
  - Processor initiates actions
  - Controller notifies processor of events, such as data availability, error condition, …
- Processor can perform other operations concurrently
- Device operation not limited by processor performance or load

Example: LCD Module

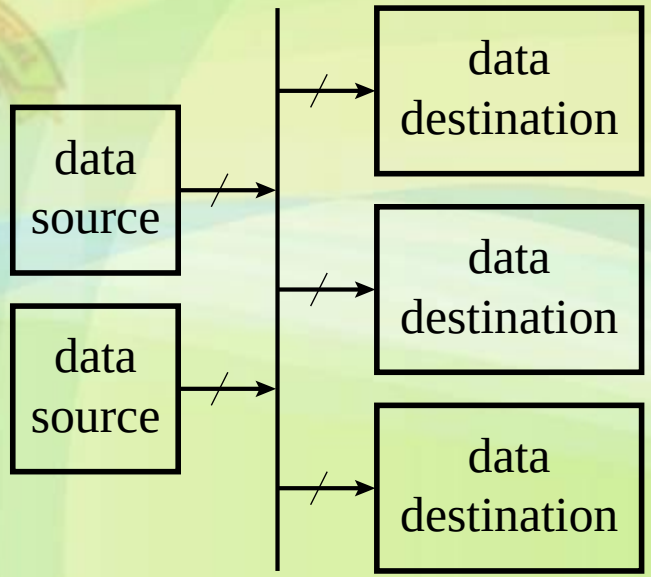
- Rectangular array of pixels
  - Row and column connections
  - Controller scans rows, activates columns
- Image or character data stored in a small memory in the controller
  - Updated by an attached processor

Direct Memory Access (DMA)

- For high-speed input or output
  - Processor writes starting address to a control register
  - Controller transfers data to/from memory autonomously
  - Notifies processor on completion/error
- Reduces load on processor
- Common with accelerators

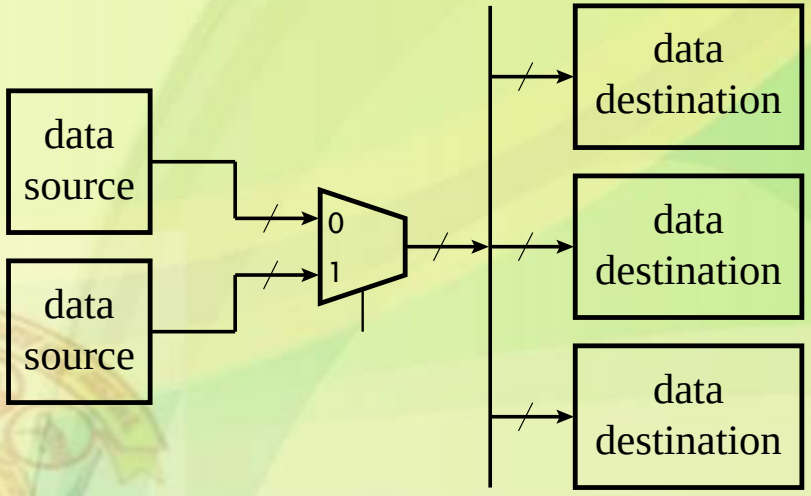
Parallel Buses

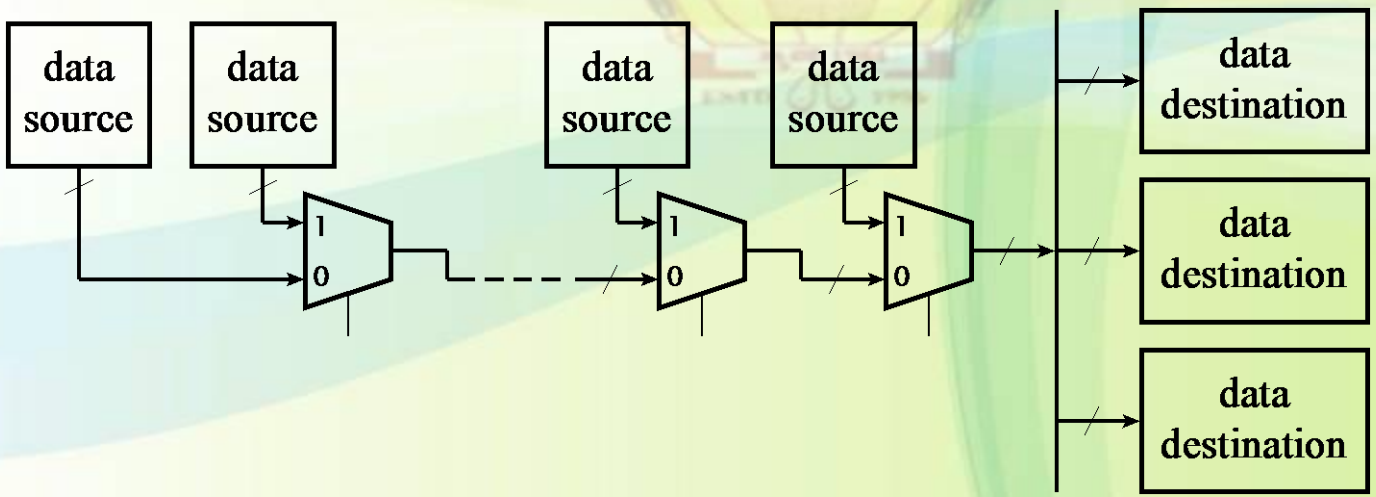
- Interconnect components in a system
  - Transfer bits of data in parallel
- Conceptual structure
  - All inputs and output connected
- In practice
  - Can't tie multiple outputs together



Multiplexed Buses

- Use multiplexer(s) to select among data sources
  - Can partition to aid placement on chip

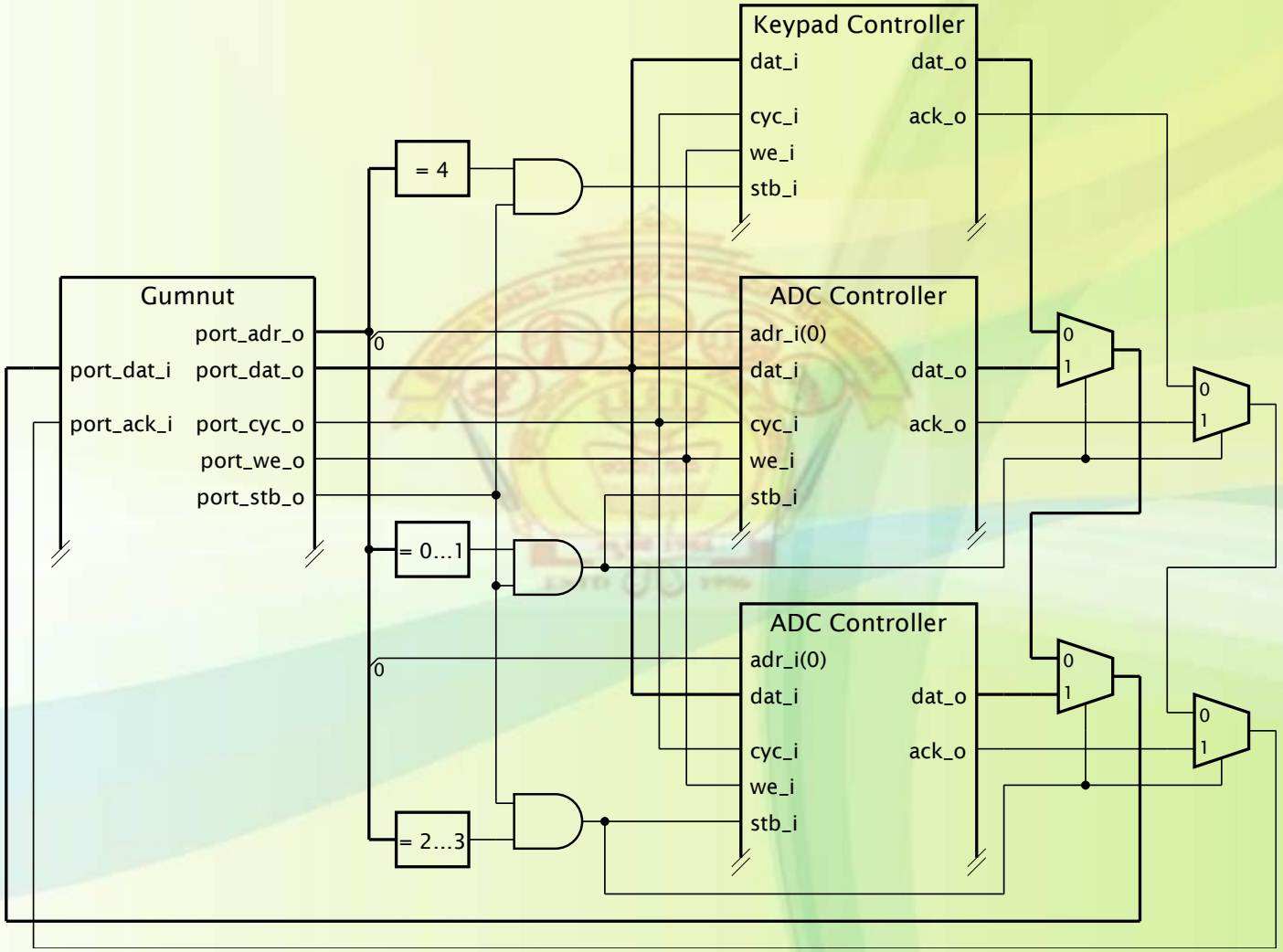




Example: Wishbone Bus

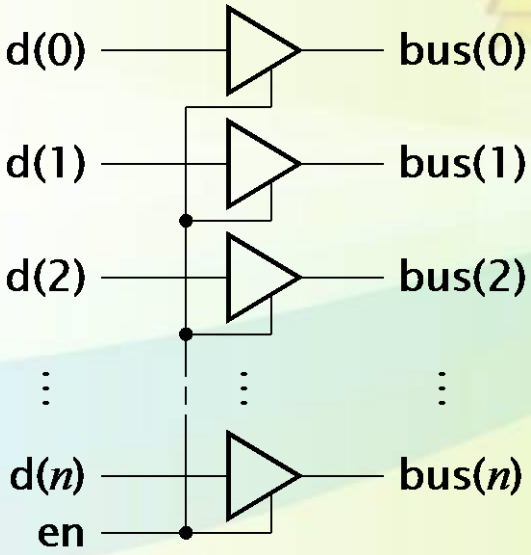
- Non-proprietary bus spec
  - OpenCores Organization
- Gumnut uses simple form of Wishbone
  - One bus for each of instruction memory, data memory, and I/O ports
  - "…_o" denotes output
  - "…_i" denotes input

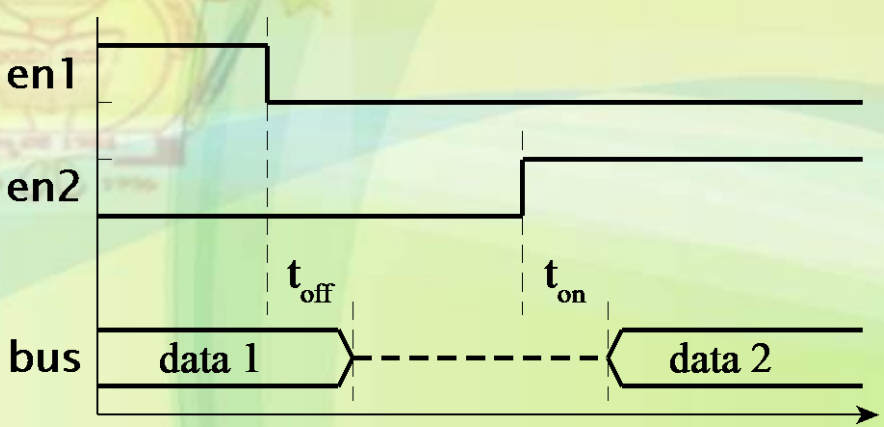
Example: Wishbone Bus



Tristate Buses

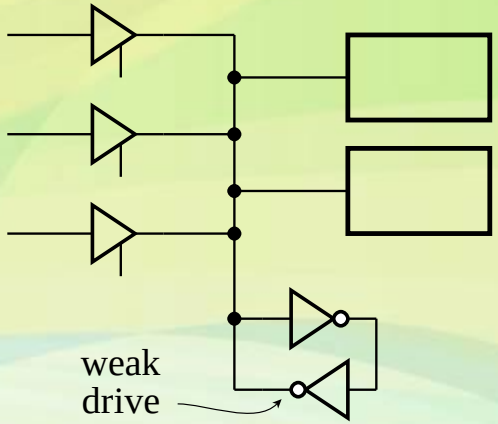
- Use tristate drivers for data sources
  - Can “turn-off” (Hi-Z) when not supplying data
- Simplified bus wiring

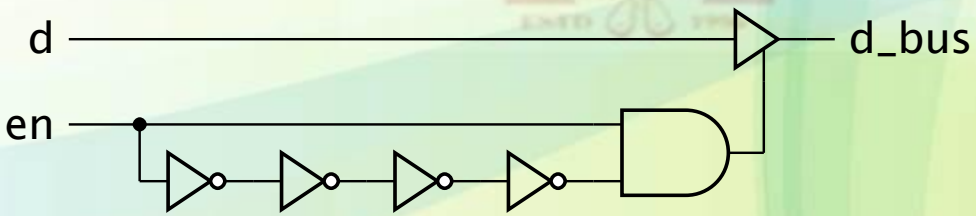




Tristate Bus Issues

- Floating bus can cause spurious switching
  - Use pull-up resistors or weak keepers
- Need to avoid driver contention
  - Dead cycle between turn-off and turn-on
  - Or delayed enable



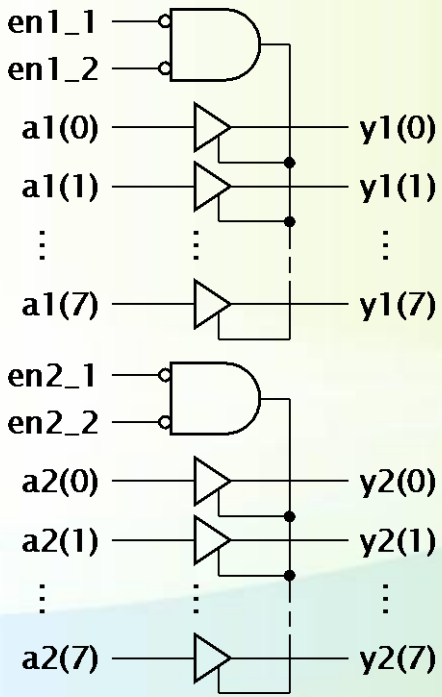


- Not all CAD tools and implementation fabrics support tristate buses

Tristate Drivers in Verilog

- Assign Z to an output to turn driver off
- Example: single-bit driver
  - `assign d_out = d_en ? d_in : 1'bz;`
- Example: multi-bit driver
  - `assign bus_o = dat_en ? dat : 8'bz;`
- Any other driver contributing 0 or 1 overrides Z value

Example: SN74x16541



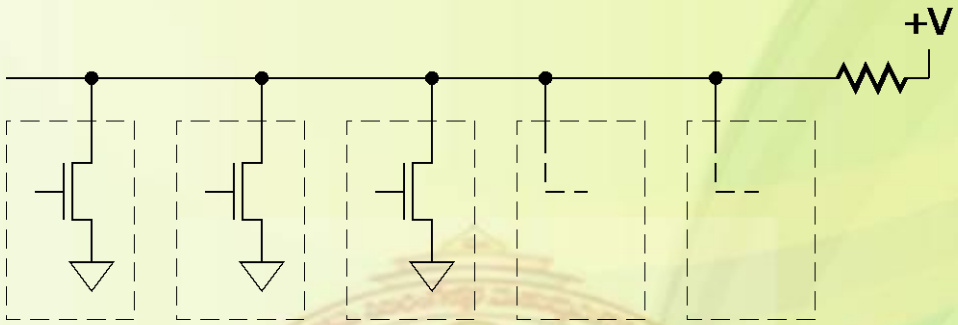
Same as wire, but indicates tristate driver

```
module sn74x16541 ( output tri [7:0] y1, y2,
                    input      [7:0] a1, a2,
                    input            en1_1,
                                     en1_2,
                                     en2_1,
                                     en2_2 );
  assign y1 = (~en1_1 & ~en1_2) ? a1 : 8'bz;
  assign y2 = (~en2_1 & ~en2_2) ? a2 : 8'bz;
endmodule
```

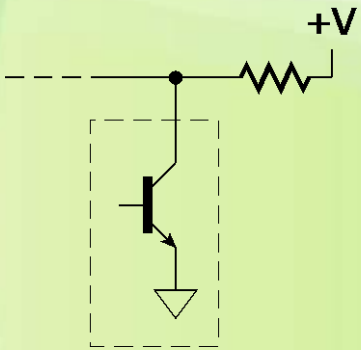
Unknown Values in Verilog

- What if two drivers are turned on?
  - One driving 0, the other driving 1
  - Resolved value is X — unknown
  - Can test for X during simulation
    - Use === and !== operators
    - C.f. == and !=, which are logical equivalence and inequivalence tests
- Z and X are not electrical logic levels
  - Notations for simulation and synthesis
  - Real logic levels are only 0 or 1

Open-Drain Buses



- Bus is 0 if any driver pulls it low
- If all drivers are off, bus is pulled high
  - Wired-AND
- Can also use open-collector drivers



Open-Drain Drivers in Verilog

- Assign 0 or 1 to model driver
- Model pull-up on open-drain bus using **`wand`** net
  - **`wand`** `bus_sig;`
  - Resolved value is logical AND of driver values

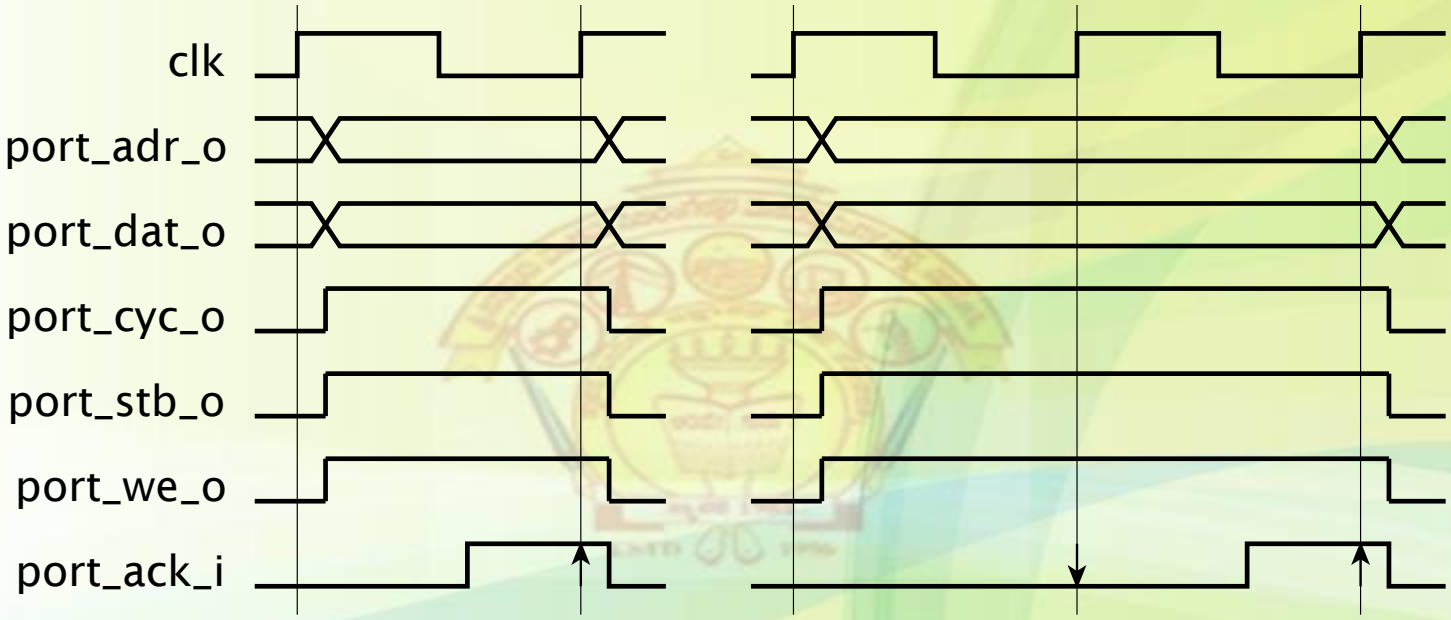
Bus Protocols

- Specification of signals, timing, and sequencing of bus operations
  - Allows independent design of components
  - Ensures interoperability
- Standard bus protocols
  - PCI, VXI, …
    - For connecting boards in a system
  - AMBA (ARM), CoreConnect (IBM), Wishbone (Open Cores)
    - For connecting blocks within a chip

Example: Gumnut Wishbone

- Minimal 8-bit subset used for I/O ports
- Signals
  - port_cyc_o: “cycle” control for sequence of port operations
  - port_stb_o: “strobe” control for an operation
  - port_we_o: write enable
  - port_ack_i: acknowledge from addressed port
  - port_adr_o: 8-bit port address
  - port_dat_o: 8-bit data output from Gumnut
  - port_dat_i: 8-bit data input to Gumnut

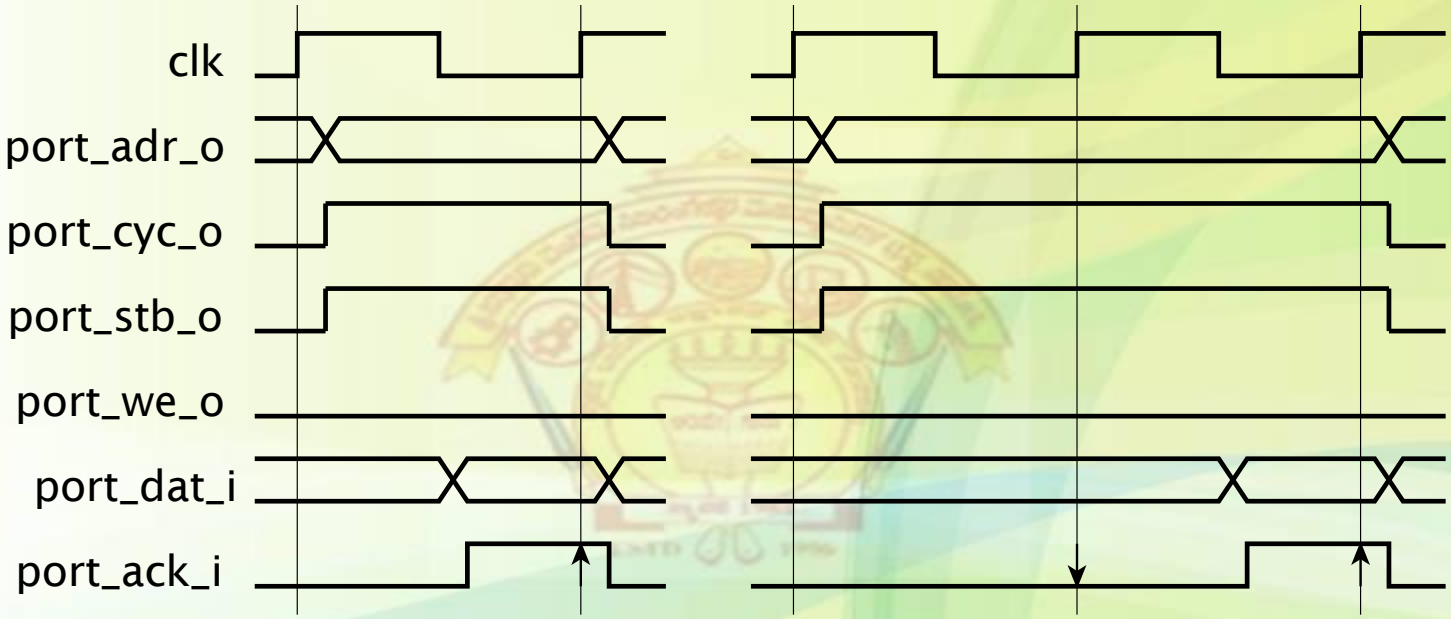
Gumnut Wishbone Write



No wait cycles

One wait cycle

Gumnut Wishbone Read



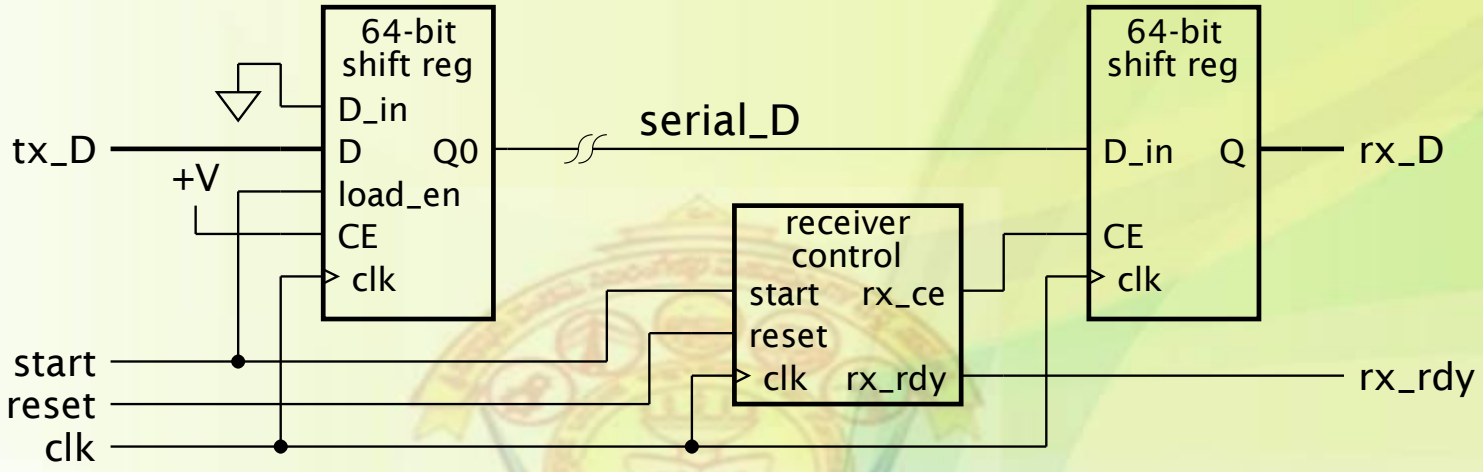
No wait cycles

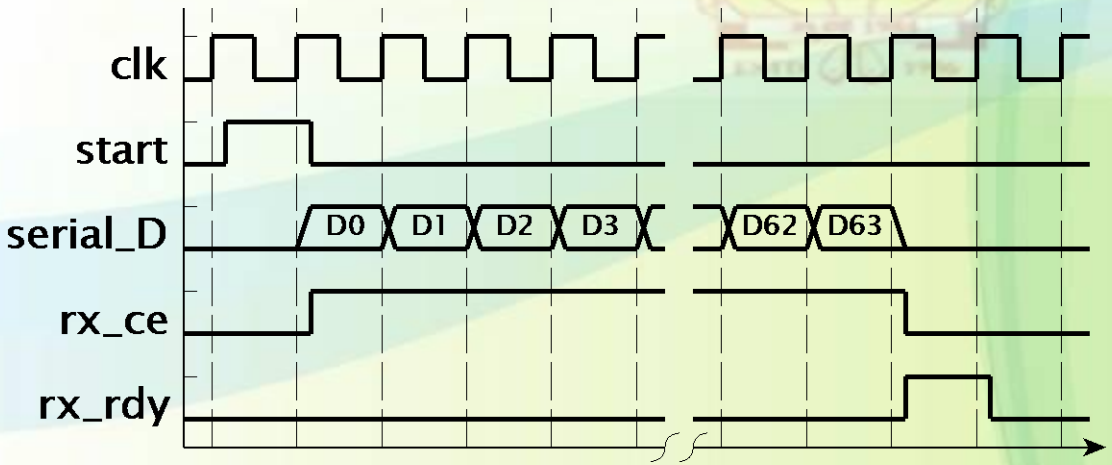
One wait cycle

Serial Transmission

- Bits transmitted one after another on a single wire
  - Can afford to optimize the wire for speed
- C.f. parallel transmission, one wire per bit
  - Requires more wires
    - Cost per wire, greater area for wiring, complexity of place & route
  - Requires more pins
    - Cost of larger package
  - Other effects
    - Crosstalk, skew, delay due to increased area
- Serializer/deserializer (serdes)
  - Converts between parallel and serial form

Example: 64-bit Serdes

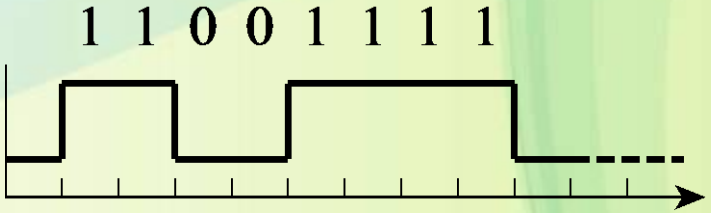




- Bit order is arbitrary, provided both ends agree
  - Often specified by standards

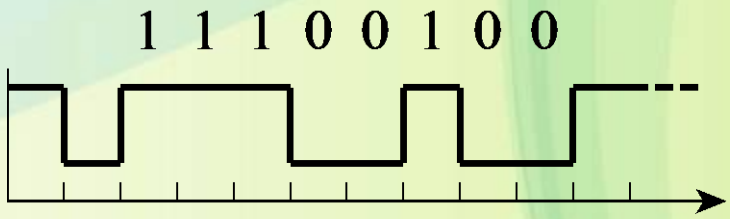
NRZ Transmission

- Non-Return to Zero
  - Just set signal to high or low for each bit time
  - No indication of boundary between bit times
  - Need to synchronize transmitter and receiver separately
    - E.g., by a common clock and control signals, as in previous example



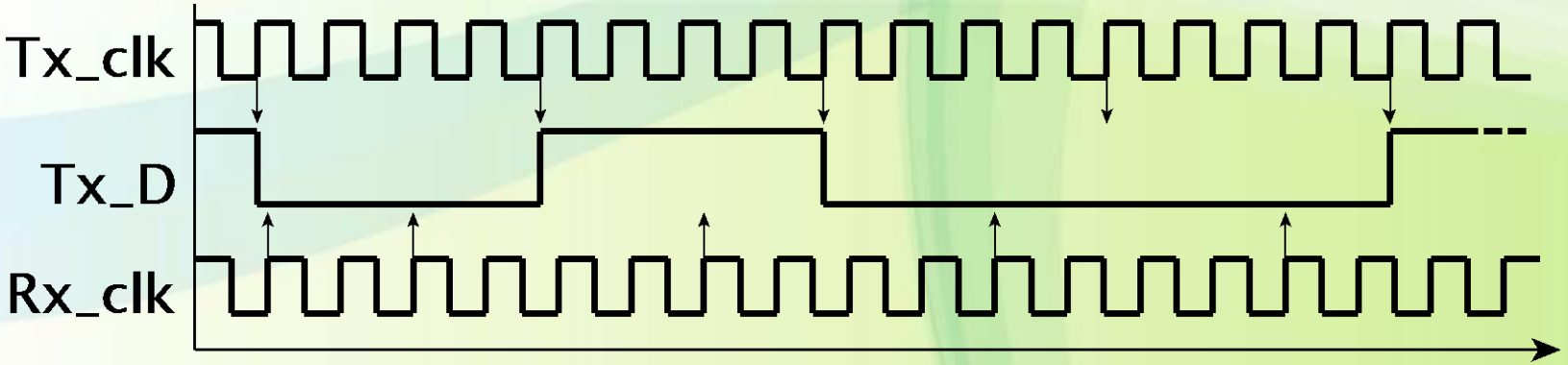
Start/Stop Bit Synchronization

- Hold signal high when there is no data
- To transmit
  - Drive signal low for one bit time (start bit)
  - Then drive successive data bits
  - Then drive signal high for one bit time (stop bit)



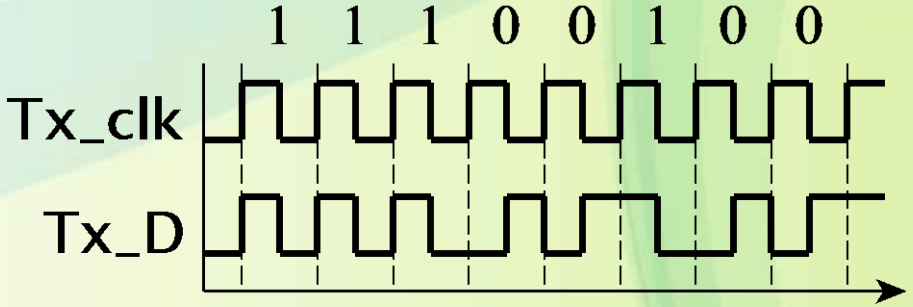
UARTs

- Universal Asynchronous Receiver/Transmitter
  - Common I/O controller for serial transmission using NRZ with start/stop bits
  - Relies on Tx and Rx clocks being approximately the same frequency



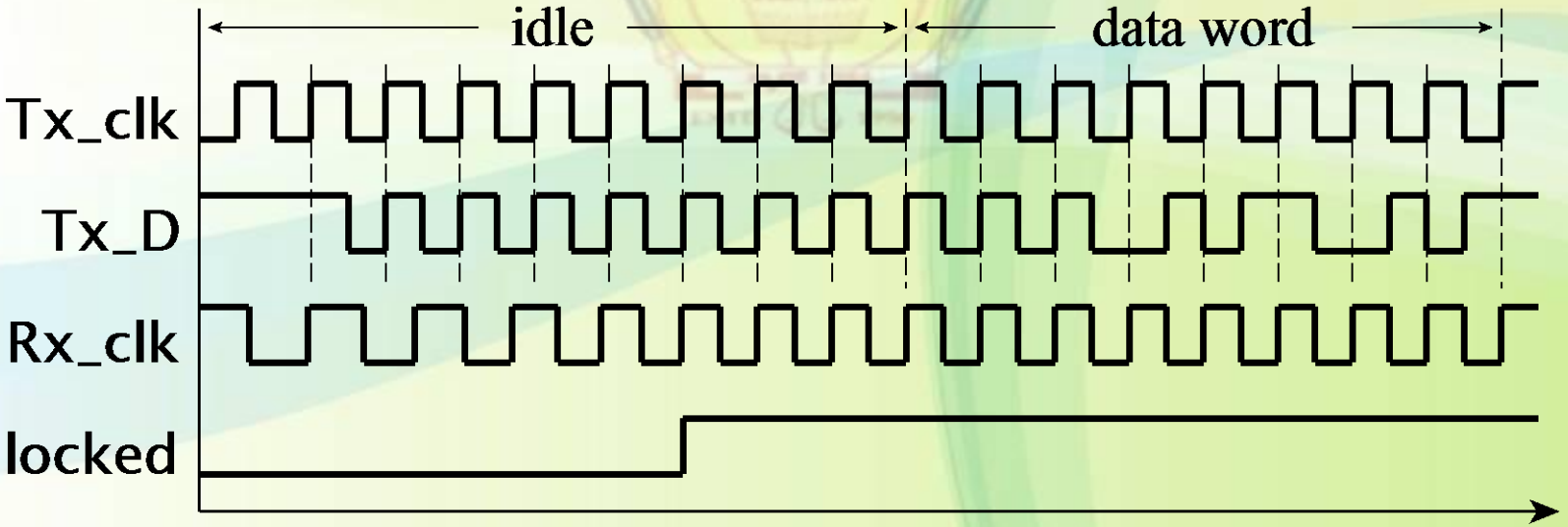
Manchester Encoding

- Combine Tx clock with Tx data
  - Ensures regular edges in the serial signal
- Example: Manchester encoding
  - Transition in the middle of each bit time
    - 0: low-to-high transition
    - 1: high-to-low transition
    - May need a transition at the start of a bit time



Clock Recovery

- Transmitter sends preamble before data
  - A sequence of encoded 1 bits
  - Serial signal then matches Tx clock
- Receiver uses a phase-locked loop (PLL) to match Rx clock to Tx clock



Serial Interface Standards

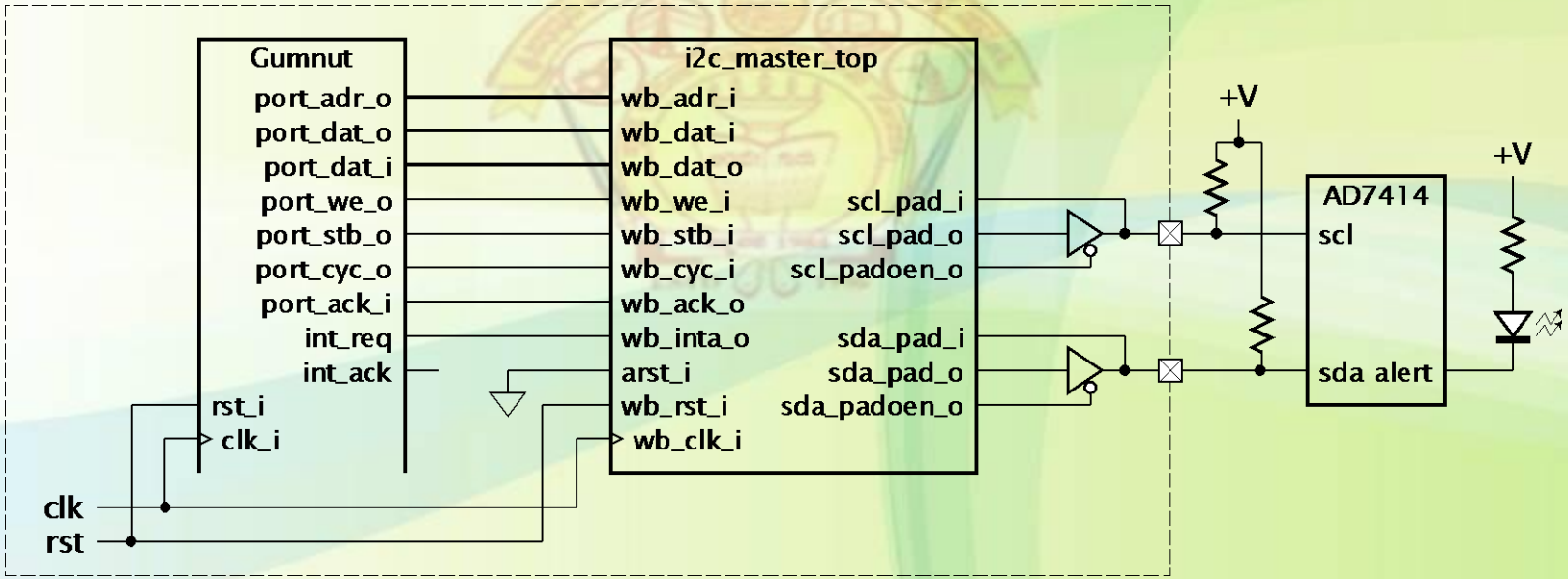
- Connection of I/O devices to computers
- Connection of computers in networks
- Use of standards reduces design effort
  - Reuse off-the-shelf components or IP
- RS-232: NRZ, start/stop bits
  - Originally for modems, now widely used for low-bandwidth I/O

Serial Interface Standards

- I^2C: Inter-Integrated Circuit bus
  - 2 wires (NRZ data, clock), open drain
  - Simple protocol, low cost, 10kb/s–3.4Mb/s
- USB: Universal Serial Bus
  - For connecting I/O devices to computers
  - Differential signaling on 2 wires
  - 1.5Mb/s, 12Mb/s, 480Mb/s, …, complex protocl
  - IP blocks available
- FireWire: IEEE Std 1394
  - 2 differential pairs (data, synch)
  - 400Mb/s, 3.2Gb/s, complex protocol

I²C Example: Temperature Sensor

- Gumnut, Analog Devices AD7414
  - I²C controller IP from OpenCores respository



I/O Software

- Use input and output instructions to access I/O controller registers
- I/O devices interact with the physical world
  - Software must respond to events when they occur
  - It must be able schedule activity at specific times or at regular intervals
  - *Real-time* behavior

Polling

- Software repeatedly reads I/O status to see if an event has occurred
  - If so, it performs the required action
- Multiple controllers
  - Software executes a polling loop, checking controllers in turn
- Advantage: simple I/O controllers
- Disadvantages
  - Processor is continually busy, consuming power
  - Delay in dealing with an event if processor is busy with another event

Polling Example

- Safety monitor in factory automation
  - Gumnut core
  - 16 alarm inputs
    - One per bit in registers at addresses 16 & 17
    - 0 $\Rightarrow$ ok, 1 $\Rightarrow$ abnormal condition
  - Temp sensor ADC at address 20
    - 8-bit binary code for °C
    - Above 50°C is abnormal
  - Alarm output at address 40
    - 0 $\Rightarrow$ ok, 1 $\Rightarrow$ ring alarm bell

Polling Example

```
alarm_in_1: equ  16       ; address of alarm_in_1 input register
alarm_in_2: equ  17       ; address of alarm_in_2 input register
temp_in:    equ  20       ; address of temp_in input register
alarm_out:  equ  40       ; address of alarm_out output register
max_temp:   equ  50       ; maximum permissible temperature

poll_loop:  inp  r1, alarm_in_1
            sub  r0, r1, 0
            bnz  set_alarm ; one or more alarm_in_1 bits set
            inp  r1, alarm_in_2
            sub  r0, r1, 0
            bnz  set_alarm ; one or more alarm_in_2 bits set
            inp  r1, temp_in
            sub  r0, r1, max_temp
            bnc  set_alarm ; temp_in > max_temp
            out  r0, alarm_out ; clear alarm_out
            jmp  poll_loop
set_alarm:  add  r1, r0, 1
            out  r1, alarm_out ; set alarm_out bit 1 to 1
            jmp  poll_loop
```

Interrupts

- I/O controller notifies processor when an event occurs
  - Processor interrupts what it was doing
  - Executes interrupt service routine
    - A.k.a. interrupt handler
  - Then resumes interrupted task
  - May enter low-power standby
- Some systems prioritize interrupt requests
  - Allow higher priority events to interrupt service of lower priority events

Interrupt Mechanisms

- Interrupt request signal
- Means of disabling/enabling interrupts
  - So processor can execute *critical regions*
- Save processor state on an interrupt
  - So interrupted task can be resumed
- On interrupt, disable further interrupts
  - Until processor has saved state
- Find the handler code for the event
  - *Vector*: address of handler, or index into table of handler addresses
- Instruction to return from handler
  - Restoring saved state

Gumnut Interrupt Mechanisms

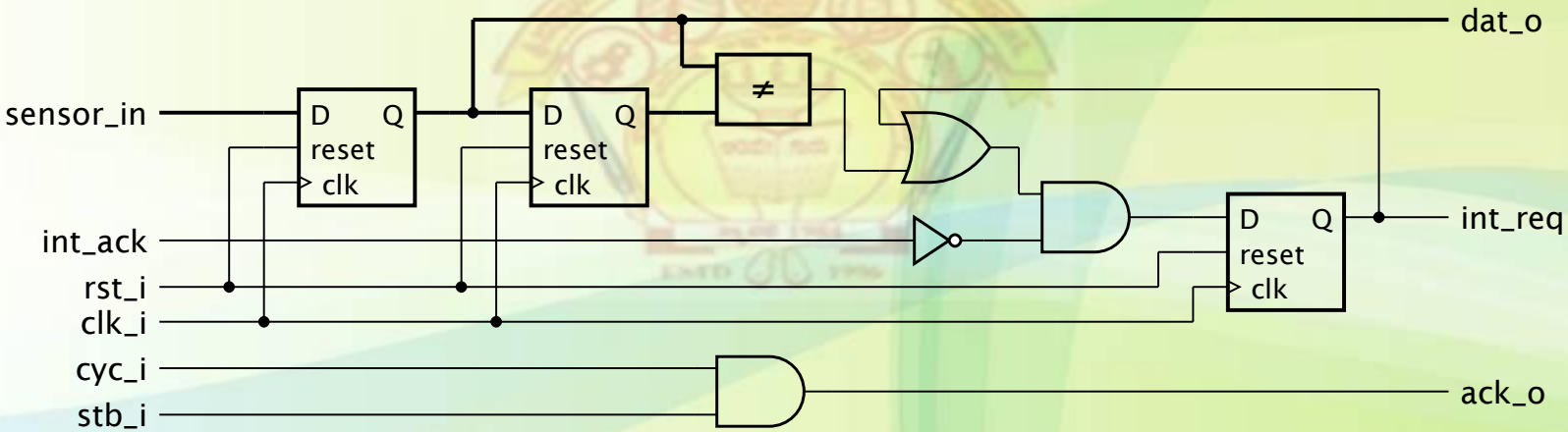
- `int_req` signal
- `disi` and `enai` instructions
- On interrupt, PC, Z, and C saved in special registers
- On interrupt, further interrupts are disabled
- Handler code starts at address 1
  - Gumnut sets PC to 1
- `reti` instruction
  - Restores PC, Z, and C from special registers, re-enables interrupts

Interrupt Acknowledgment

- Process may not respond immediately
  - But must tell controller when it does
  - Controller then deactivates request
    - To avoid multiple interrupts for one event
- Processor *acknowledges* the request
  - E.g., `int_ack` signal on Gumnut
  - Alternative: reading a status register

Example: Sensor Controller

- 8-bit input from sensor
  - Interrupt request on change of value



Example: Sensor Handler

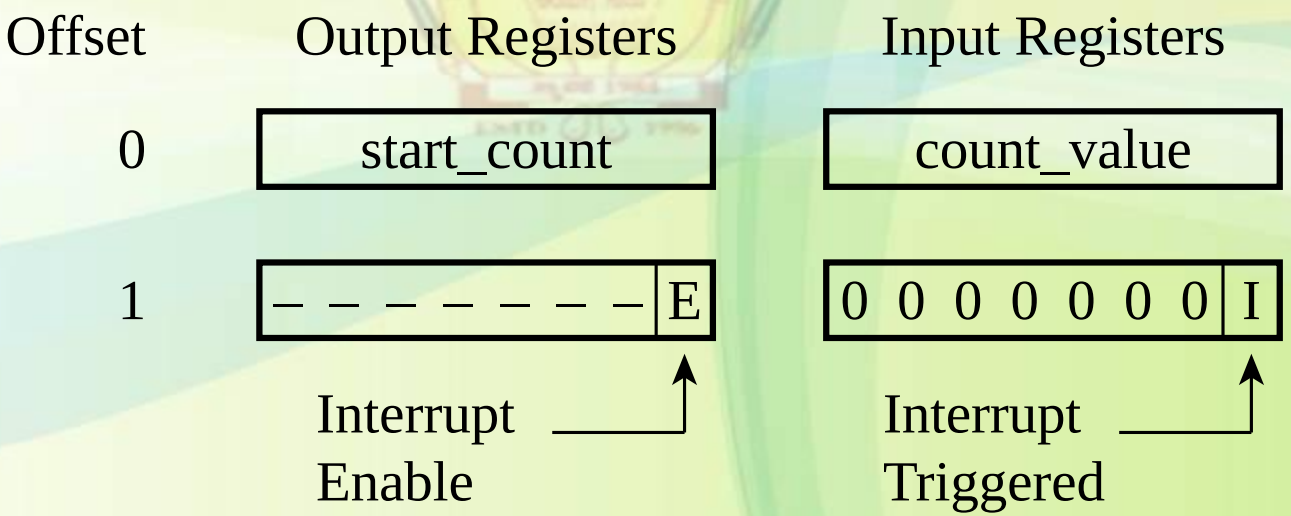
```
                data
saved_r1:       bss     1

                text
sensor_data:    equ     0           ; address of sensor data
                                    ; input register

                org     1
                stm     r1, saved_r1
                inp     r1, sensor_data
                ...                 ; process the data
                ldm     r1, saved_r1
                reti
```

Timers

- Real-time clock (RTC)
  - Generates periodic interrupts
  - Uses a counter to divide system clock
  - Control register for divisor
- Interrupt handler can perform periodic tasks
  - E.g., activate next digit of a scanned display

Example: RTC for Gumnut

- 10µs timebase, divided by a down counter
  - Initial count loaded from a register
  - Interrupt triggered on count value = 0



Real-Time Executives

- Control program
  - A.k.a. real-time operating system (RTOS)
  - Timing based on a real-time clock
  - Schedules execution of tasks
    - In response to interrupts and timer events
- Can also manage other resources
  - Memory allocation
  - Storage (file system)
  - Use of I/O controllers
  - Use of accellerators

Example: Gumnut Executive

- RTC based at address 16
- Calls `task_2ms` every 2ms

```
;;; ------------------------------------------------------
;;; Program reset: jump to main program
                text
                org     0
                jmp     main
;;; ------------------------------------------------------
;;; Port addresses
rtc_start_count:        equ     16 ; data output register
rtc_count_value:        equ     16 ; data input register
rtc_int_enable:         equ     17 ; control output register
rtc_int_status:         equ     17 ; status input register
```

Example: Gumnut Executive

```
;;; -----------------------------------------------------
;;; init_interrupts:    Initialize 2ms periodic interrupt, etc.

                        data
rtc_divisor:            equ     199     ; divide 100kHz down
                                        ; to 500Hz
rtc_int_flag:           bss     1


                        text
init_interrupts:        add     r1, r0, rtc_divisor
                        out     r1, rtc_start_count
                        add     r1, r0, 1
                        out     r1, rtc_int_enable
                        stm     r0, rtc_int_flag
                        ...     ; other initializations
                        ret
```

Example: Gumnut Executive

```
;;; -----------------------------------------------------------
;;; Interrupt handler
                        data
int_r1:                 bss     1 ; save location for
                                  ; handler registers

                        text
                        org     1

int_handler:            stm     r1, int_r1      ; save registers
check_rtc:              inp     r1, rtc_status  ; check for
                                                ;  RTC interrupt
                        sub     r0, r1, 0
                        bz      check_next
                        add     r1, r0, 1
                        stm     r1, rtc_int_flag ; tell main
                                                 ; program
check_next:             ...

int_end:                ldm     r1, int_r1 ; restore registers
                        reti
```

Example: Gumnut Executive

```
;;; -----------------------------------------------------------
;;; main program

                text
main:           jsb     init_interrupts
                enai
main_loop:      stby
                ldm     r1, rtc_int_flag
                sub     r0, r1, 1
                bnz     main_next
                jsb     task_2ms
                stm     r0, rtc_int_flag
main_next:      ...
                jmp     main_loop
```

- Note: task_2ms not called as part of interrupt handler
  - Would slow down response to other interrupts

Summary

- Transducers: sensors and actuators
  - Analog-to-digital and digital-to-analog coverters
- Input and output devices
- Controllers
  - Input, output, control, and status registers
  - Autonomous controllers
- Buses: multiplexed, tristate, open-drain
  - Bus protocols: signals, timing, operations

Summary

- Serial transmission
  - NRZ, embedded clock
- Real-time software
  - Reacting to I/O and timer events
  - Polling, interrupts
- Real-time executives

Queries ...?