### RTL Design Process

<table>
<thead>
<tr>
<th>Step 1: Capture behavior</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Capture a high-level state machine</em></td>
<td>Describe the system’s desired behavior as a high-level state machine. The state machine consists of states and transitions. The state machine is “high-level” because the transition conditions and the state actions are more than just Boolean operations on single-bit inputs and outputs.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 2: Convert to circuit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2A</td>
<td><em>Create a datapath</em></td>
</tr>
<tr>
<td>Create a datapath</td>
<td>Create a datapath to carry out the data operations of the high-level state machine.</td>
</tr>
<tr>
<td>2B</td>
<td><em>Connect the datapath to a controller</em></td>
</tr>
<tr>
<td>Connect the datapath to a controller</td>
<td>Connect the datapath to a controller block. Connect external control inputs and outputs to the controller block.</td>
</tr>
<tr>
<td>2C</td>
<td><em>Derive the controller’s FSM</em></td>
</tr>
<tr>
<td>Derive the controller’s FSM</td>
<td>Convert the high-level state machine to a finite-state machine (FSM) for the controller, by replacing data operations with setting and reading of control signals to and from the datapath.</td>
</tr>
</tbody>
</table>

Sources: TSR, Katz, Boriello, Vahid, Perkowski
Example: Simple data encryption/decryption device

- $B = 1$, set offset $O = I[0:31]$
- $B=0$ $e=1$: encrypt mode: output $J = I+O$
- $B=0$ $e=0$: decrypt mode: get $J=I-O$
Hot Water Detector

Create an alarm system that sets alarm=1 when the average temperature of four consecutive samples CT meets or exceeds a threshold WT. Signal clr=1 disables the alarm.
Example: Design from “C” code

```
Inputs: byte a, byte b, bit go
Outputs: byte gcd, bit done
GCD:
while(1) {
    while(!go);
    done = 0;
    while ( a != b ) {
        if( a > b ) {
            a = a - b;
        } else {
            b = b - a;
        }
    }
    gcd = a;
    done = 1;
}
```

Sources: TSR, Katz, Boriello, Vahid, Perkowski
Fibonacci Lookup Table

- Design a lookup table 256 x 256 bit that stores Fibonacci #'s:
  - $F_n = 0$ if $n = 0$, $1$ if $n = 1$, $F_{n-1} + F_{n-2}$ otherwise ($n<256$)
Finish HLSM Design

- Design an 8-bit counter using RTL:
  - When input $E = 1$, it counts even numbers ($0,2,4,6,..$) and when $E = 0$, it counts odd numbers ($1,3,5,7,..$).
  - When input $CLR = 1$ and $E=1$, then it clears the output to 0; if $CLR=1$ and $E=0$, it sets output to “00000001”.
  - If you were initially counting even(odd) numbers, and $E$ flips, then the output changes to the nearest greater odd (even) value.
RTL Summary

• Datapath and Control Design
• RTL Design Steps
  1. Define the high level state machine
  2. Create datapath
  3. Connect datapath with control
  4. Implement the FSM
• Timing analysis – critical path in more complex circuits
  – Watch out for all possible long paths (e.g. datapath to FSM, FSM control logic, datapath logic etc)
CSE140: Components and Design Techniques for Digital Systems

Memory

Tajana Simunic Rosing
Welcome to CSE 140!

• **Where we are going today:**
  – RTL examples, Memory

• **Upcoming:**
  – HW6 due today, HW7 assigned
  – Midterm #3 in CENTR 101 on 3/18 at 11:30am

• **CAPEs are out!!!** [https://cape.ucsd.edu/students/](https://cape.ucsd.edu/students/)
  – Your feedback is very important to us! We read your feedback carefully and use it for future courses. 😊
  – At 90% CAPE response rate, I will drop the lowest HW grade!

• **Bonus points for the class:**
  – Bonus take home final exam, posted on Monday, 3/20, due Saturday, 3/25
  – Worth 2% of the course grade

• **Regrades:**
  – Grades posted Th, regrade due the following Th; updated grades posted on TED by the weekend
  – Check your grades in TED, let us know if anything is missing
    • iClicker scores updated as of week 8

• **Discussion session** on Friday at 2pm!
Memory: basic concepts

• Stores large number of bits
  – $m \times n$: $m$ words of $n$ bits each
  – $k = \log_2(m)$ address input signals
  – or $m = 2^k$ words
  – e.g., 4,096 x 8 memory:
    • 32,768 bits
    • 12 address input signals
    • 8 input/output data signals

• Memory access
  – r/w: selects read or write
  – enable: read or write only when asserted
  – multiport: multiple accesses to different locations simultaneously
• Traditional ROM/RAM
  – ROM
    • read only, bits stored without power
  – RAM
    • read and write, lose stored bits without power
• Distinctions blurred
  – Advanced ROMs can be written to
    • e.g., EEPROM
  – Advanced RAMs can hold bits without power
    • e.g., NVRAM

Write ability and storage permanence of memories, showing relative degrees along each axis (not to scale).
Comparing RAM

- Register file
  - Fastest
  - But biggest size
- SRAM
  - Fast (e.g. 10ns)
  - More compact than register file
- DRAM
  - Slowest (e.g. 20ns)
    - And refreshing takes time
  - But very compact
  - Different technology for large caps.

Sources: TSR, Katz, Boriello, Vahid, Perkowski
Caches and CPUs

- Servers: Level 1 (L1), L2 & L3 cache on chip
- Embedded: L1, L2 on chip

Intel Xeon Server

Raspberry Pi 3

Sources: TSR, Katz, Boriello, Vahid, Perkowski
Raspberry Pi 3 – Memory Architecture

- Broadcom BCM2837 SoC
  - CPU: Quad-core Cortex-A53: L1 and L2 cache
  - GPU: VideoCore IV® Processor: exclusive memory system
  - Main Memory: 1GB RAM: Shared by CPU and GPU
Tightly coupled memory (TCM) = SRAM organized as main memory
Random Access Memory (RAM)

- RAM – Readable and writable memory
  - Logically the same as register file
    - RAM just one port; register file two or more
  - RAM vs. register file
    - RAM is larger
    - RAM stores bits using a bit storage vs. FFs
    - RAM implemented on a chip in a square – keeps longest wires (hence delay) short
  - Why random?
    - When RAM first came out, the comparison was with sequential read of magnetic tape 😊
• Similar internal structure as register file
  – Decoder enables appropriate word based on address inputs
  – rw controls whether cell is written or read
  – Let’s see what’s inside each RAM cell
“Static” RAM cell
- 6 transistors (recall inverter is 2 transistors)
- Writing this cell
  - *word enable* input comes from decoder
  - When 0, value *d* loops around inverters
    - That loop is where a bit stays stored
  - When 1, the *data* bit value enters the loop
    - *data* is the bit to be stored in this cell
    - *data’* enters on other side
    - Example shows a “1” being written into cell
Static RAM (SRAM) - reading

- “Static” RAM cell - reading
  - When rw set to read, the RAM logic sets both data and data’ to 1
  - The stored bit d will pull either the left line or the right bit down slightly below 1
  - “Sense amplifiers” detect which side is slightly pulled down
Dynamic RAM (DRAM)

• “Dynamic” RAM cell
  – 1 transistor (rather than 6)
  – Relies on large capacitor to store bit
    • Write: Transistor conducts, data voltage level gets stored on top plate of capacitor
    • Read: Just look at the value of \( d \)
    • Problem: Capacitor discharges over time
      – Must “refresh” regularly, by reading \( d \) and then writing it right back
Comparing Memory Types

- Register file
  - Fastest
  - But biggest size
- SRAM
  - Fast
  - More compact than register file
- DRAM
  - Slowest
    - And refreshing takes time
  - But very compact
  - Different technology for large caps.
SRAM timing

- CE'
- R/W'
- Adrs
- Data

From SRAM → From CPU

read  write  time

Sources: TSR, Katz, Boriello, Vahid, Perkowski
DRAM timing

- CE'
- R/W'
- RAS'
- CAS'
- Adrs
- Data

Sources: TSR, Katz, Boriello, Vahid, Perkowski
DRAM Page mode access

Sources: TSR, Katz, Boriello, Vahid, Perkowski
Ram variations

- **PSRAM: Pseudo-static RAM**
  - DRAM with built-in memory refresh controller
  - Popular low-cost high-density alternative to SRAM

- **NVRAM: Nonvolatile RAM**
  - Holds data after external power removed
  - Battery-backed RAM
    - SRAM with own permanently connected battery
    - writes as fast as reads
    - no limit on number of writes unlike nonvolatile ROM-based memory
  - SRAM with EEPROM or flash
    - stores complete RAM contents on EEPROM or flash before power turned off

Sources: TSR, Katz, Boriello, Vahid, Perkowski
Extended data out DRAM

- Improvement of FPM (full page mode) DRAM
- Extra latch before output buffer
  - allows strobing of cas before data read operation completed
- Reduces read/write latency by additional cycle
(S)ynchronous and Enhanced Synchronous (ES) DRAM

- SDRAM latches data on active edge of clock
- Eliminates time to detect *ras/cas* and *rd/wr* signals
- A counter is initialized to column address then incremented on active edge of clock to access consecutive memory locations
- ESDRAM improves SDRAM
  - added buffers enable overlapping of column addressing
  - faster clocking and lower read/write latency possible

Sources: TSR, Katz, Boriello, Vahid, Perkowski
Rambus DRAM (RDRAM)

• More of a bus interface architecture than DRAM architecture
• Data is latched on both rising and falling edge of clock
• Broken into 4 banks each with own row decoder
  – can have 4 pages open at a time
• Capable of very high throughput
RAM Example: Digital Sound Recorder

- **Behavior**
  - Record: Digitize sound, store as series of 4096 12-bit digital values in RAM
    - We'll use a 4096x16 RAM (12-bit wide RAM not common)
  - Play back later from RAM
**RAM Example: Digital Sound Recorder**

*Record behavior*

Keep local register $a$
- Stores current address, ranges from 0 to 4095
- Create state machine that counts from 0 to 4095 using $a$
- For each $a$
  - Read analog-to-digital conv: $ad_{ld}=1$, $ad_{buf}=1$
  - Write to RAM at address $a$: $Ra=a$, $Rrw=1$, $Ren=1$
RAM Example: Digital Sound Recorder

Play behavior

Local register:  a (12 bits)

- V
  - a=0
  - ad_buf=0
  - Ra=a
  - Rrw=0
  - Ren=1

- W
  - a<4095
  - da.ld=1
  - a=a+1
  - a=4095

- X
  - da.ld=1

Create state machine that counts from 0 to 4095; for each a:
  - Read RAM
  - Write to digital-to-analog conv.
  - Note: Must write d-to-a one cycle after reading RAM, when the read data is available on the data bus

Sources: TSR, Katz, Boriello, Vahid, Perkowski
• **FIFO Queue** (first-in-first-out)
  – Write at the back: *push*, Read at the front: *pop*
  – Treat memory as a circle

• Common uses:
  – Computer keyboard
    • Pushes pressed keys onto queue; Meanwhile pop and send to computer
  – Digital video recorder
    • Pushes frames onto queue; Meanwhile pops frames, compresses them, and stores them
  – Routers
    • Pushes incoming packets onto queue; Meanwhile pops packets, processes destination information, and forwards each packet out over appropriate port

Sources: TSR, Katz, Boriello, Vahid, Perkowski
Two conditions have front=rear need FSM to detect:
- Full: No pushes until a pop
- Empty: No pops until a push

Use Register file for storage

Implement Rear and front with up counters:
- rear as RF’s write address, front as read address
Read-Only Memory – ROM

- Memory that can only be read from
  - Data lines are output only
- Advantages over RAM
  - Nonvolatile
  - Low power
  - Compact

Let $A = \log_2 M$

Sources: TSR, Katz, Boriello, Vahid, Perkowski
ROM Example: Digital Telephone Answering Machine

- Record the outgoing announcement
  - When $\text{rec}=1$, record digitized sound in locations 0 to 4095
  - When $\text{play}=1$, play those stored sounds to digital-to-analog converter
ROM Example: Digital Telephone Answering Machine

- High-level state machine
  - Once $rec=1$, begin erasing flash by setting $er=1$
  - Wait for flash to finish erasing by waiting for $bu=0$
  - Execute loop that sets local register $a$ from 0 to 4095, reading analog-to-digital converter and writing to flash for each $a$

Sources: TSR, Katz, Boriello, Vahid, Perkowski
Non-volatile memory yesterday and today

- **Erasable Programmable ROM (EPROM)**
  - Uses “floating-gate transistor” in each cell
  - Programmer uses higher-than-normal voltage so electrons tunnel into the gate
    - Electrons become trapped in the gate
    - Only done for cells that should store 0
    - Other cells will be 1
  - To erase, shine ultraviolet light onto chip
    - Gives trapped electrons energy to escape
    - Requires chip package to have window

- **Electronically-Erasable Programmable ROM (EEPROM)**
  - Programming similar to EPROM
  - Erasing one word at a time electronically

- **Flash memory**
  - Like EEPROM, but large blocks of words can be erased simultaneously

- **EEPROM & FLASH are in-system programmable**
Non-volatile memory going forward

• A new class of data storage/memory devices
• Emerging NVMs have exciting features:
  – Non-volatile like Flash (~ 10 years)
  – Fast access times (~ SRAM)
  – High density (~ DRAM)
• NVM *blurs the distinction* between
  – Memory *(fast, expensive, volatile)* &
  – Storage *(slow, cheap, non-volatile)*
• Key issues:
  – Slow writes, low endurance, costly and complex manufacturing
1T-1C FeRAM

• Similar in construction to DRAM
  – Both cell types include one capacitor and one access transistor
  – DRAM cell capacitor use a linear dielectric; FeRAM includes ferroelectric material, typically lead zirconate titanate (PZT)
  – Writing is accomplished by applying a field across the ferroelectric layer by charging the plates on either side of it, forcing the atoms inside into the "up“/logic “1” or "down“/logic "0“

• Advantage:
  – No need for refresh – 99% lower power than DRAM
  – Similar performance to DRAM

• Disadvantage:
  – It is unclear how it will scale as materials stop being ferroelectric at small sizes (now produced in 130nm)
  – Reads are destructive – data has to be rewritten
STT-RAM: Spin-Transfer Torque RAM

- The spin torque direction of electrons to flip a bit in a magnetic tunneling junction (MTJ)

- **Advantages:**
  - High endurance & fast reads

- **Disadvantages:**
  - *Write energy*: large current needed to reorient the magnetization for most commercial applications;
  - *Asymmetric write*: Writing a “1” needs much more time and energy than writing a “0”
Domain Wall Memory (DWM)

- Similar to STT-RAM structure
- **Advantage:**
  - needs only one tunneling barrier and fixed layer → area savings
- **Disadvantages:**
  - complexity of design, read/write delay due to sequential access

Sources: TSR, Katz, Boriello, Vahid, Perkowski
Shift-based DWM

- Writes by shifting data of one of the two fixed layers with the desirable direction comp
- **Advantage**: faster writes than a traditional DWM
- **Disadvantage**: cost and manufacturing complexity

1-bit DWM is fast  
Multi-bit DWM is area efficient, but needs extra latency for shifting

Sources: TSR, Katz, Boriello, Vahid, Perkowski
PCM: Phase Change Memory

Phase Change Memory (PCM)
- Flips a bit by changing the state of material
- Crystalline (SET) and amorphous (RESET) phase

Advantages:
- better scalability than other emerging technologies

Disadvantages:
- Slow writes
- Low endurance ($10^7$ writes)

Candidate for DRAM replacement
ReRAM: Resistive RAM

- Two types: Access-based (1T-1R) and crossbar ReRAM (1T-nR)

- **Access-based ReRAM (1T-1R)**
  - A dielectric, which is normally an insulator, can conduct with sufficiently high voltage

- **Advantage:**
  - Very fast reads and writes ~ 20ns
  - Very high density

- **Disadvantage:**
  - Limited endurance (10^5 writes)

Sources: TSR, Katz, Boriello, Vahid, Perkowski
Crossbar ReRAM

- **Crossbar ReRAM (1T-nR)**
- **Advantage:**
  - Highly scalable
  - Can be implemented at the top of the chip with 3D architecture
  - Very low energy consumption
  - Low cost (possible replacement for Flash)
- **Disadvantage:**
  - Much slower than 1T-1R \(\sim\) us

Sources: TSR, Katz, Boriello, Vahid, Perkowski
Crossbar RRAM Applications

Crossbar 1T-nR

Data Center – Block Oriented Data Storage
- Low latency ($10^{-5}$ s), high bandwidth, low energy
- Native 3D compatible array architecture
- Alternatives: SSD, DRAM (disk cache), hard disk

1T-1R

Nodes – XiP Code and Random Access Data
- Low latency ($10^{-8}$ s), high bandwidth, low energy
- Alternatives: embedded NOR, discrete NOR

NVDRAM Comparison
NVMs Comparison

- **STT-RAM**: SRAM cache replacement
- **PCRAM**: DRAM main memory and storage
- **ReRAM**: NAND flash, embedded NOR flash

<table>
<thead>
<tr>
<th>Features</th>
<th>SRAM</th>
<th>eDRAM</th>
<th>STT-RAM</th>
<th>PCRAM</th>
<th>ReRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density Speed</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Very high</td>
<td>Very high</td>
</tr>
<tr>
<td></td>
<td>Very Fast</td>
<td>Fast</td>
<td>Fast for read; slow for write</td>
<td>Slow for read; very slow for write</td>
<td>Slow for read/write</td>
</tr>
<tr>
<td>Dynamic Power</td>
<td>Low</td>
<td>Medium</td>
<td>Low for read; very high for write</td>
<td>Medium for read; high for write</td>
<td>Medium for read; high for write</td>
</tr>
<tr>
<td>Leakage Power</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Non-volatility</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Sources: TSR, Katz, Boriello, Vahid, Perkowski
Summary

• Memory hierarchy
  – Needs: speed, low power, predictable

• Cache design
  – Mapping, replacement & write policies

• Memory types
  – ROM vs RAM vs NVM

• NVM
  – Many new technologies that are still maturing
  – Excellent target for big data and energy-efficient applications

Sources: TSR, Katz, Boriello, Vahid, Perkowski