# Lecture 2

- CHDL or HDL is a textual language for designing hardware circuits.
- Circuits can be specified via a diagram schematic or a textual CHDL.
- Diagrams are easier for the beginner but not very scalable or useful.
- CDHLs offer better scalability and support formal methods.
- Circuits are very similar to functions. Black box, internal computation, produce outputs.
- Give circuits a name in Hydra. This type of 'black box' definition is similar to abstraction in imperative programming languages.
- Multiplexor in primitives: mux1 c a b = or2 (and2 (inv c) a) (and2 c b).
- Think of the multiplexor order ar counting from 0 to highest ed mux2 (0,1) p q r s = q (second value).
- Demultiplexors take an control input and a data input and produce 2 outputs. The control selects which output to put the data on.
- Demux2 and Mux2 can be built by combining demux1s or mux1s
- Tuples used where there is no logical grouping and indexing doesn't make sense.
- Words used when there is a logical sense of numbering and ordering.
- Words of Tuples and Tuples or Words are allowed.

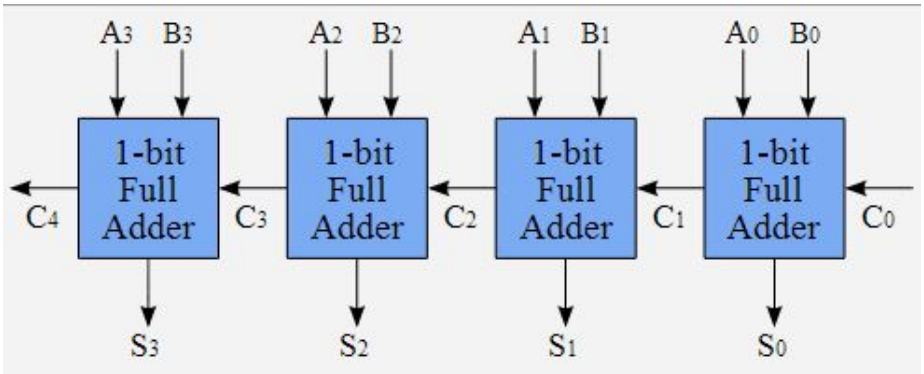
# Lecture 3

- No feedback loops in combinational logic.
- Make sure that inputs and outputs are known BEFORE implementation.
- Efficiency is tied in with a circuits cost model. Not necessarily the number of components.
- It is generally better to come up with a better algorithm.
- In VLSI regularity is better than random optimization.

1. Synthesis from truth table.
2. Synthesis from algebraic logic
3. ''Sum of cases" technique.
4. Conditionals and cases.

5. Using encodings.
6. Defining building blocks
7. Standard patterns

- ''Sum of cases" technique is a way of truth table synthesis. (mapping inputs to outputs)
- Nested multiplexors are used for conditionals and cases.
- Cases correspond to $2^k$ cases given a k-bit opcode. Use demultiplexors in these circumstances.
- Half adder - A sum and a carry. Full adder a 2 bit sum and a carry.
- Ripple adders are combinations of the 'full adder' building block




- Efficiency should always be done in reference to cost model.
- Optimization only plays a part later, it should not interfere with synthesis or it could become counterproductive.
- Cost models place a cost on a circuit. The best circuit is the one with the lowest cost.
- Examples of cost models include number of components (not useful nowadays as we have integrated circuits) and path depth (for combinational circuits).
- For k inputs. There are $2^k$ lines in its truth table and $2^{(2^k)}$ boolean functions of k inputs.
- General logic gates (LUT). Important in reconfigurable circuits. p = lut2 (a,b,c,d) x y
- Gate delay is the time it takes for a signal to become valid once it has reached the gate.

- Difference between invalidity and incorrectness.
- For each signal x in the circuit, the time at which x becomes stable is called t(x). This is the path depth of x
- Inputs to x are denoted a I(x).
- Gate delay of a gate = gate delay of previous gates and the gates own delay.

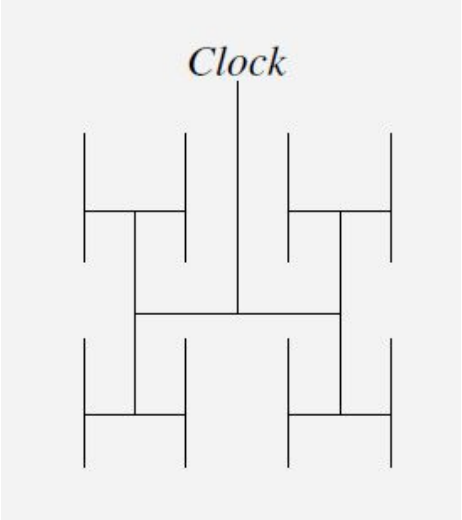
$$t(x) = d + \max_{y \in I(x)} t(y)$$

- Critical path depth is the maximum path depth of all of its outputs. This can be used to determine the 'speed' of the circuit.

# Lecture 4

- Hazards, incorrect output based on varying gate delays (ie and2 x (inv x) results in a 1 value while (inv x) suffers gate delay).
- A sequential circuit may have flip flops and feedback loops.
- Sequential circuits perform computations through time.
- Flip-flip is basic memory (hydra def - dff :: Clocked a -¿ a -¿ a)
- Clocked used a Bool is simply not rich enough.
- Real systems have a reset mechanism to put flip-flips into a valid state.
- System must ensure a valid flip-flip value (no mid-voltages).
- Flip-flops update on clock tick. Output continuous but only updates when clock pulses 1.
- Abstract clock and Physical clock differences. Abstract cannot exist in real life. Components treat a 0 to 1 transition as a tick.
- Synchronous model is used to prvent hazards. It boasts simple behavior and satisfies some constraints.
- Synchronous if:
  1. Contains logic gates and flip flops.
  2. Every flip flop is connected to a global clock.
  3. Every clock tick reaches each flip flop at the same time.
  4. Each feedback loop goes via a flip flop.
  5. Inputs to a circuit remain stable during a clock cycle.

- The restrictions on this mean that you cannot do combinational logic on clock input, you cannot introduce state via combinational feedback loops, and the circuit may malfunction if inputs are not stable for long enough.
- State can be created by using flip flops or feedback loops in combinational logic (difficult to reason).
- A synchronous circuit is a sequential circuit that behaves like a finite state automation.
- Synchronous trade off. Restricted design chouce but much easier to reason the circuit.
- Clock is an implicit input (as all dffs have the same clock ALWAYS).
- Clock tick is a point in time. Clock cycle is an interval.
- Clock skew is when a clock tick does not reach flip flops at the same time (such as in a fanout or if there are longer wires etc).
- The floor plan of the clock is normally laid out well in advance. See example H-tree. Known as the clock distribution.




- Circuits can have 'global' state in the synchronous model (values of all flip flops).
- Circuit behaves as a single finite state automation.
- Clock removes concern for intermediate values, hazards and gate delays (as they are all dealt with in the synchronous model).
- reg1 component like a dff but with option ld signal that loads/ignores input value after clock tick accordingly.

- reg1 works by multiplexing the input to a dff depending on the value of ld (obviously involves a feedback loop).

# Lecture 5

- list structure for words similar to haskell notation ie. (x:xs) = [a,b,c,d,e], x = a and b = [b,c,d,e]
- equally the ++, !! and length operators can be used like in haskell.
- field w i k takes the value of w starting at position i for k elements.
- The type of a circuit describes its input. Such as how it's represented, how many inputs/outputs it takes/produces, parameter size and building blocks (if any).
- You may use circuit generators to save making every component by hand. Eg. rippleAdd, a generic n-bit ripple adder that works for any word size.
- This is not software to do the operation, it is still a real circuit.
- Hydra has 4 'times'
  1. Model Transformation Time
  2. Circuit Generation Time
  3. Simulator Compilation Time
  4. Runtime
- Always use type a rather than Bool, its a hydra thing...
- bitslice2 can be used so that [a] -¿ [a] -¿ [(a,a)]
- mux1w is like a multiplexor but outputs words rather than single bits.
- general n-bit reg an wlatch (wlatch doesn't take a load signal).
- Modern RISC and CISC architectures use an indexed set of registers R0, R1, R2 etc which are implemented in a circuit called the register file.
- Memory locations are usually bytes while registers contain words. Registers are few in comparison to memory (but much faster).
- A dual-port register file provides separate input from output. Some register files have three ports (one in and 2 out).
- How to handle address inputs. A tree of multiplexors and demultiplexors is needed for the addresses.
- Look over register file recursive definitions.

## Lecture 6

- A test bench takes readable input and produces readable output. It converts to and from the actual signals and formats the output.
- Lots of hydra test bench definitions. 'getbit', 'getbin' and 'gettc'
- Additional hydra functions 'fanout', 'shl', 'bitslice2', 'orw'
- An ALU is a combinational circuit. It is typically an adder with augmented functions to support other functions (eg bit shifting, subtraction, incrementing).
- The critical path of the ALU should be very small.
- Multiplication, division etc are performed over a series of clock ticks (by using sequential algorithms).
- Sequential functions may operate as a machine language program, using normal processing registers or using a dedicated sequential circuit with its own registers (functional unit).
- A functional unit is a sequential black box circuit with its own specific registers to perform a task like multiplication. It receives an input start and outputs an output signal 'ready'.
- By using a functional unit (which is sequential), we can accept it may take a few clock cycles which means the global clock doesn't have to be slowed down.
- If it were done by combinational logic the path depth would be so great the clock speed would have to be slowed down.
- Use the shift and add algorithm to do multiplication. The inputs to x*y would be x y and start, the outputs would the be result and ready.

## Lecture 7

- Number of patterns in combinational logic; sum of cases, truth tables, fold, scan, map.
- Design patterns are higher order functions. They take one or more circuit specifications as parameters.
- The pattern defines how to connect up these given circuits in a regular pattern.
- Pattern definition similar to ordinary circuit specification except that: It uses recursion to decompose groups of signals, it uses abstract circuits, rather than specific circuits, its type may include building block circuits.

- map, map2, mapn used as in haskell.
- fold as well. and2/or2 operate by using left folds.
- ascanr is like fold but returns a list (including all intermediate values

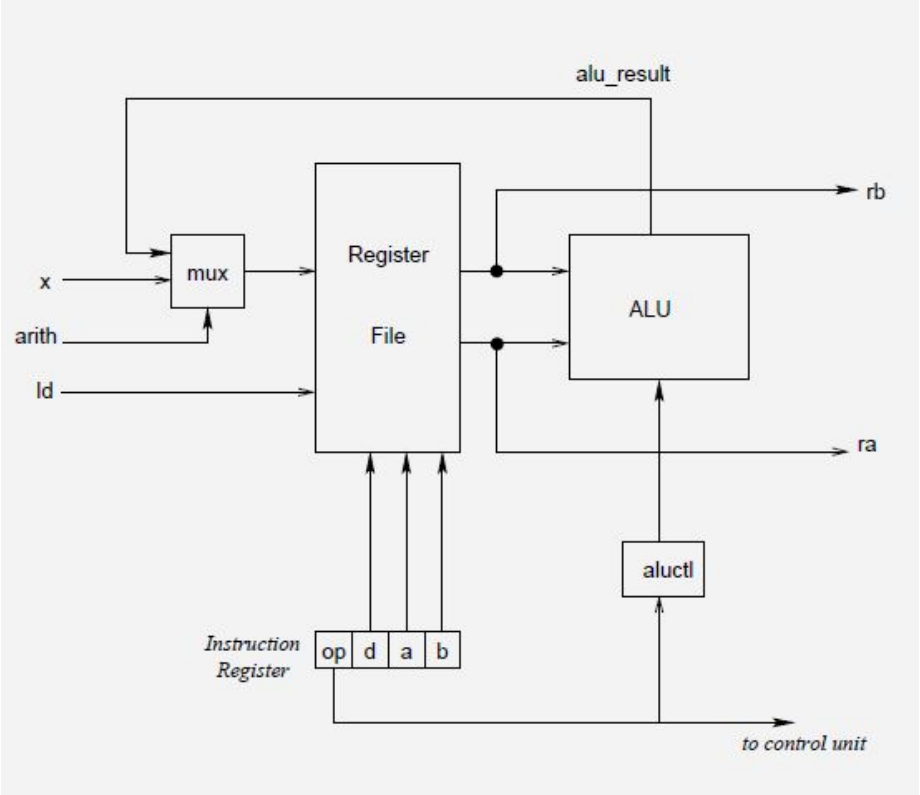
## Lecture 8

- FSA (finite state automation) has a set of states, a stream of inputs (and state conversion function). It is used to recognize if the grammar of an input is correct.
- FSAs need to be more general for practical applications. For example the new state transition function could produce a new output as well as defining a new state to enter.
- In an unencoded representation you could use a dff for each state. If exactly one flip flop contains a 1 then it is in that state.
- In an unencoded representation the no. of states = no. state flip flops.
- In an encoded representation, each possible state is represented by a unique binary number held in a register.
- In an encoded representation, no. of flip-flips = $\log_2$(no. of states)
- For the shift and add algorithm we must distinguish between what to put in the registers when start = 1 AND what to put in the registers when start $\neq$ 1 but a multiplication is still taking place.
- When start = 1 x & rx are both k bits, y i k bits, ry is 2k bits (it needs padding) and prod is 2k bits. rx and ry are set to the inputs on signals x & y and prod is set to 0.
- rx and ry are shifted right and left accordingly so that the algorithm is always examining the same bits.

## Lecture 9

- Sigma 16 is a 16-bit RISC style architecture. No bytes/words. Everything is 16-bit. 16 registers. Based on a simplified version of MIPS.
- RRR format (two operands in the register file. The result goes into a register)
- RX format (one operand is a register, the other is a memory address).
- Memory addresses are constants, they are the combination of the displacement and the index/base.

- RRR takes 4 inputs. The opcode op, destination d, and source addresses sa & sb.
- RX takes 4 inputs. The RX opcode (MUST be f), a destination register d, an index register sa and the secondary opcode sb.
- RRR (one word) RX (two words, 4 4 bit fields and a 16-bit displacement).
- Instructions can often be implemented without a feedback loop.




- The datapath contains the combinational logic for calculations.
- The control generates all of the control signals needed for the datapath
- Similar design complexities, the datapath focuses more on structure whereas the control focuses on behavior.
- Different design techniques (datapath designed directly as a circuit, control as an algorithm). The separation helps clarify timing.
- Design of the datapath (all the logic and registers needed for calculations) will logically create a set of control signals that operate it.

- There is a naming convention for control signals. ctl, subsystem, specific. Eg ctl_rf_ld.
- The datapath contains registers, computational systems and interconnections.
- It has two inputs, the control signals and a data word (from either memory system or the DMA input controller).
- It outputs the bus to memory an I/O (it also outputs the states of the registers).
- The datapath can be broken down into further subsystems (registers, ALU, busses etc).
- Introduce an instruction register and program counter. For RX instructions an adr register is needed for the second word

## Lecture 10

- adr, pc and ir needed in the datapath to keep track of program execution. ALU needed to perform arithmetic.
- The datapath contains the processor's rigiters.
- Aim

## Lecture 13

- Latency is the time required for an operation. There is memory, instruction and communication latency.
- Throughput is the amount of work performed per unit of time (eg. memory accesses per second).
- For overall performance throughput is important.
- For (re)solving bottlenecks, latency is often important.
- Amdahl's law observes that the total speed up of the system may be less than the speed up of a part of the system depending on how much time is spend in said part of the system.
- Example of Amdahl's law. If part A of a system can be sped up from 30s to 15s and is 30% of the system process (the other part takes 70s) then the increase is 85s/100s which is 1.17 overall speed up factor.
- Moore's law observes manufacturing trends. It states that the number of transistors per chip will double every two years (or some other number of years).

- Moore's law is based on the fact that integrated circuit fabrication technology makes gradual improvements which allows more components to be placed on chips.
- Moore's law is not about speed. It is about chip density. The only way to increase speed now is to add more processing cores to a chip.
- Performance should only really be measured in terms of its 'wall clock' time to run a piece of software.

$$T = \text{ni} \cdot \text{cpi} \cdot \text{spc}$$

$$or$$

$$T = \frac{\text{ni} \cdot \text{cpi}}{\text{cps}}$$

- Where T = execution time in seconds, ni = no. instructions in order to execute the program, cpi = average no. clock cycles per instruction, spc = seconds per clock cycle, cps = cycles per second.
- Reducing one of these factors does not necessarily reduce T.
- In practice, the critical path will be the path from a register in a register file, through the ALU and back to the register file.
- The M1 control algorithm will place whatever value is on the memory bus into the destination register. Even if the value is incorrect (for instance if it takes several clock cycles to compute).
- More sophisticated solutions include using pipelining stages, interleaved memory and superscalar memory.
- The M1 circuit has no interrupts, memory protection or virtual memory. No concurrency. Long-running instructions must execute until completion while the rest of the processor waits.
- Pipelining is the application of parallelism to the steps in executing an instruction.
- The basic application of pipelining is to