CSSE2010 Diagram Notation

Table of contents

What was read . . . 2
The course ships no notation guide for these diagram families . . . 2
Figure sources the slides cite . . . 3
Gate symbols and logic diagrams . . . 4
Block and MSI symbols . . . 5
Timing and waveform diagrams . . . 7
State diagrams . . . 7
State tables . . . 8
Block and datapath diagrams . . . 9
Register and memory layout diagrams . . . 10
AVR architecture figures . . . 11
Truth tables and function tables . . . 12
Boolean and textual notation . . . 12
Colour and annotation . . . 13
Inconsistencies . . . 14
Not established by the source . . . 16
See also . . . 17

The drawing conventions this course actually uses, inferred from its own materials. This is the reference every later diagram in CSSE2010 should be checked against.

Scope. This note covers the diagram families that the two existing notation notes do *not*: gate symbols and logic diagrams, timing/waveform diagrams, state diagrams and state tables, block and datapath diagrams, register/memory layout diagrams, and the AVR architecture figures. It also records the algebraic and textual notation the figures depend on. For circuit-schematic rules (device IDs, gate letters, pin numbers, junction dots, power blocks, IO-board input/output pentagons) see circuit-schematics[1]; for the 74-series pinouts and their schematic symbols see device-pinouts[2]. Those two are not repeated here.

[1] courses/csse2010/circuit-schematics.pdf
[2] courses/csse2010/device-pinouts.pdf

What was read

An honest statement of the sample, because most of what follows is inferred from drawings rather than stated in prose.

Guides — all three read in full (9 of 9 pages):

- *CSSE2010/CSSE7201 Device Pinouts and Symbols* (v4), pp. 1–2
- *CSSE2010/CSSE7201 — Guide to Drawing Circuit Schematics* (v2.3, Semester One 2016), pp. 1–2
- *I/O Board User Manual*, pp. 1–5

Lecture decks — 221 of 246 pages:

Deck	Pages read	of
L01 Intro and Binary Numbers	1–20	21
L02 Intro to Logic Gates	1–24	24
L03 Binary Arithmetic	8–17	17
L04 Combinational Logic	1–19	19
L05 Flip-Flops	4–19	19
L06 Shift Registers	3–17	17
L07 Counters	3–17	17
L08 State Machines	2–21	21
L09 ALU and Control Unit	3–22	22
L10 AVR Introduction	3–19	19
L11 ISA Part 1	3–28	28
L12 ISA Part 2	2–20	22

Unread: L01 p21, L03 pp1–7, L05 pp1–3, L12 pp21–22 — title, outline and admin pages in every case except L03 pp1–7 (number-representation recap, no new diagram family). Labs, past papers and the course profile were not sampled at all.

Citation format below: "L04 s10" means lecture deck 04, slide 10. Slide numbers printed on the slides match the PDF page numbers in every deck checked.

The course ships no notation guide for these diagram families

The three guides are *hardware* documents. Between them they settle schematic layout, device labelling, pin numbering and the physical IO board — nothing about gate-symbol family, waveform drawing, state diagrams, block diagrams or memory layout. The *Device Pinouts and Symbols* sheet comes closest: it has a "Logic Diagram Symbol" column (p. 1–2) that shows the gate shapes and the flip-flop/mux/adder block symbols, but it gives no rules, only examples.

So: **everything in the sections below except the schematic rules is inferred from the drawings**, and every entry is cited so you can check the inference.

Figure sources the slides cite

This matters more than it looks. L01 s8 says **"No textbook required"** — yet four different figure sources are credited across the semester, each bringing its own drawing style. There is no single house style.

Source, quoted as printed	Where	What it supplies
`[Figure from Mano, "Digital Design", 3rd edition]`	L03 s16, L04 s4, L04 s10, L05 s17, L08 s3	Ripple-carry adder, 4-to-1 mux circuit, sequential-circuit block diagram
`[Mano, Digital Design, 3rd ed.]` (short form)	L06 s6, L06 s8, L06 s9	4-bit register, 4-bit shift register
`Images taken from Instructor Resources of Digital Fundamentals, Thomas L. Flyod, Pearson Educations Inc., 2015` (misspelling in the original)	L07 s4, s5, s7	Shift-register type diagrams, digital-system application block diagram
`[Figure from Tanenbaum, "Structured Computer Organization"]` — no edition given	L09 s8	Full ALU internal logic diagram
`[From ATmega324A datasheet]`	L11 s14	AVR data and program memory maps
`Some datasheet extracts follow` (prose, not a caption)	L10 s9, covering s10–s11	AVR CPU architecture, ATmega324A chip block diagram
`Source: Google Images`	L01 s4	Photo collages only, not line art

Tanenbaum is also cited in prose twice as the source of *terminology*, not figures: `(Tanenbaum calls this Program Status Word)` (L11 s19) and `These addressing mode names are as per Tanenbaum "Structured Computer Organization"` (L12 s16).

Consequence for a new diagram: the block-diagram idiom the course leans on most is Mano's (five citations across four decks). If you redraw a block diagram and want it to look native, match Mano, not Floyd. Floyd appears in exactly one deck. See Inconsistencies — this is not a settled question.

Gate symbols and logic diagrams

Symbol family

Distinctive-shape symbols throughout — no IEC rectangles anywhere. AND is a flat-backed body with a semicircular nose; OR has a concave back and a pointed nose; NOT is a triangle with a bubble on its tip; NAND/NOR/XNOR are the base shape with a bubble on the output; XOR/XNOR add a second concave arc set back from the input edge.

Evidence: the definitional table at L02 s6–s8 (all seven gates), then every working diagram in the course — L02 s14, s18, s19, s22, s23; L03 s15; L04 s10, s14; L05 s10, s11, s13; L07 s3, s12; L09 s6, s7, s8, s10; L11 s6. Also the "Logic Diagram Symbol" column of *Device Pinouts and Symbols* p. 1–2.

Consistency: **holds without exception across all 221 deck pages and all three guides.** The course never names the family; the identification as ANSI/IEEE Std 91 distinctive-shape is mine, from the shapes.

Inversion bubble

A small **open** circle at the symbol boundary, always. On outputs it makes NAND/NOR/XNOR/NOT (L02 s6–s8). It is also legitimate on **inputs**, to express a De Morgan equivalent: L02 s22 draws OR-with-inverted-inputs NAND, and AND-with-inverted-inputs NOR, with bubbles sitting on the input leads at the body edge. Active-low control pins carry it too (74HCT157 `ENABLE`, *Device Pinouts* p. 2). *Consistency:* every occurrence, throughout.

Flow direction and wiring

- **Inputs on the left, outputs on the right.** Stated in words, twice: L02 s14 ("Inputs are conventionally on left" / "Outputs are conventionally on right") and the *Guide to Drawing Circuit Schematics* p. 1. Held in every logic diagram sampled.
- **Orthogonal routing only** — wires run horizontally and vertically, turning at right angles; no diagonal segments in any figure. (L02 s14, s18; L04 s14; L05 s13; L11 s6.)
- **Junction dot = connection. Crossing with no dot = no connection.** L02 s14 labels both cases explicitly on the same figure ("Indicates Electrical Connection", "No Connection"). Restated as error 2 of the schematics guide p. 2. *Consistency:* the single most-repeated drawing rule in the course.
- One line weight for all signals in a gate-level diagram. Thick lines appear only in datapath block diagrams (see below).

Labelling inside a logic diagram

- **Gate-type names are written above the symbol only in the definitional table** (L02 s6–s8: "AND", "OR", "NAND", "NOR", "XOR", "XNOR"; the NOT symbol on s6 is left unnamed). In every *working* diagram the shape alone carries the function — no type label. (L02 s14, s18, s19; L03 s15; L04 s14; L09 s6.)
- **Signal names**: single uppercase letters for primary inputs (`A`, `B`, `C`, `D`, `E`), and `X`, `M`, `F` or `Z` for outputs (L02 s6–s8, s14, s18; L03 s15; L04 s14). Complements carry an overline ($\bar{A}$).
- **Intermediate signals are labelled on the wire, immediately at the gate's output**, with the expression they carry: `AB`, `AC`, `AB + AC`, `B + C`, `A(B + C)` (L02 s19; L05 s16; L06 s5), or written above the wire ($\bar{A}BC$, `ABC`, `ABC`, `ABC` — L02 s18).
- **Numbered gates:** where a figure needs to refer to individual gates it puts a bare arabic numeral inside the gate body (L02 s18: inverters 1–3, ANDs 4–7, OR 8). **Seen once.** Extrapolate with caution.
- **Signal rails:** when one variable feeds many gates, it is run as a vertical rail crossing the whole figure, tapped with junction dots, and labelled at the top of the rail (L02 s18 — rails `A B C` $\bar{A}$ `B C`; L04 s14, the 3-to-8 decoder, same idiom). Two instances, two decks.
- **Sub-figure labels** are lowercase letters in parentheses, centred under each variant: `(a) (b) (c) (d)` (L02 s22; L05 s14, s15).

Block and MSI symbols

Multiplexer — trapezoid, wide edge left

Long parallel edge on the **left** carrying the data inputs stacked top-to-bottom (`D0 … D3`, or bare `0/1/2/3` when the inputs are anonymous); short parallel edge on the **right** with the single output; **select inputs enter through the bottom sloping edge**, labelled outside the body (`S1 S0`, `SEL`, `DIRN`, `F1 F0`).

Evidence: L04 s9, s10, s11, s12; L06 s12; L09 s6, s7, s10; L11 s6; *Device Pinouts* p. 2 (74HCT157, with `SEL` and `ENABLE` on the bottom edge). *Consistency:* **five decks plus the guide — solid.**

Decoder — trapezoid, wide edge right

The mirror flare: short edge on the **left** with the n inputs (`A B C`), long edge on the **right** with the 2 outputs (`D0 … D7`). (L04 s13.) *Consistency:* **one figure only.** The orientation is unmistakable in it, but it is a single instance.

Adder — trapezoid with a + in the body

Long edge left with the two operand groups and the carry-in at the top corner; short edge right with the sum group and the carry-out. A large + sits in the middle of the body. (L04 s7; *Device Pinouts* p. 2, 74HCT283.) The Floyd figure instead labels its adder block `A Σ / B C_out` (L07 s7) — see Inconsistencies.

D flip-flop — plain rectangle

`D` inside at the top-left, `Q` inside at the top-right, clock entering the **lower-left** edge, `Q` under `Q` when the complement is needed. The **triangular wedge (>) at the clock input means edge-triggered**; no wedge means a level-triggered latch; **a bubble in front of the wedge means falling-edge**, no bubble means rising-edge. L05 s14 states all four cases explicitly as figures (a)–(d) with the caption "Triangle indicates edge-triggered (therefore flip-flop)".

Evidence: L05 s5, s6, s14; L06 s4, s10, s11, s12; L07 s3, s8, s10, s11, s12; *Device Pinouts* p. 1–2. *Consistency:* the rectangle-with-wedge is universal; only the clock **label** varies (see Inconsistencies).

Asynchronous set/clear, when shown, are written `SET` above and `CLR` below inside the box (L07 s3, s12) and **may be omitted entirely when unused** — *Device Pinouts* p. 1 says so in words: "(Set and clear inputs can be omitted if not needed)". Active-low asynchronous inputs are marked by overlining the name (`PRE̅`, `CLR` — L05 s13) or by a bubble on the pin (*Device Pinouts* p. 1, 74HCT74). L06 s4 states the rule in prose: "The SET and CLR inputs are typically active-low".

Register — rectangle with a name plate

Rectangle with the word `Register` written **above** the box, input pin names down the left edge, output pin names down the right, a plain vertical line joining the first and last name of a contiguous range (meaning "and all the ones between"), enable name centred at the bottom inside, clock wedge on the left edge. (L06 s15, the 4-stage 8-bit queue.) **One figure.**

ALU

Two shapes, both used, both in the same deck — see Inconsistencies. The **notched trapezoid** (wide edge up, a V-notch cut into the top edge, operands entering the top, result leaving the bottom, `ALU` lettered in the body) is the more common: L09 s15, s16, s20; L10 s10. The **plain rectangle** with `ALU` centred and heavy arrows on all four sides is used at L09 s5 and s9.

Timing and waveform diagrams

See timing-diagrams[3] for what a timing diagram *means*; this is how it is drawn.

- **Time runs left to right.** Where the axis is marked at all it is a short solid horizontal arrow labelled `Time` placed under or to the right of the waveform block (L04 s15, s18). Most figures omit it (L04 s16, s17; L05 s6, s8).
- **One horizontal track per signal, stacked vertically.** The signal name sits to the **left** of its track, **followed by a colon**: `Input:`, `Output:`, `D:`, `CLK:`, `Q:`, `A:`, `B:`. (L04 s15, s16, s17; L05 s6, s8.) *Consistency:* every timing figure sampled.
- **Two horizontal dashed reference lines bound each track** — upper = logic 1, lower = logic 0. Where they are labelled, the labels read `Logic 1` and `Logic 0` and sit at the end of the rails (L04 s15, s18). In most figures the rails are drawn but left unlabelled (L04 s16, s17; L05 s6, s8).
- **The waveform is a heavy solid black line**, noticeably heavier than the dashed rails, riding on the lower rail for 0 and the upper for 1.
- **Idealised transitions are vertical** (zero rise/fall time): L04 s15, s16, s17; L05 s6, s8. **Sloped edges appear only in the one figure whose subject is real timing** — L04 s18, "Gates Aren't Perfect … The Reality of Timing", where the slope is the point.
- **Input waveforms enumerate every input combination exactly once, in equal-width intervals.** L04 s16 states it: "All combinations of inputs are covered". For two inputs the pattern is A = `0 0 1 1`, B = `0 1 0 1` across four equal intervals (L04 s16, s17).
- **Measured intervals** are marked by dropping vertical dashed lines from the two instants and drawing a **red double-headed arrow** between them, labelled in **blue** (`Propagation delay`, `Fall time`, `Rise time`). L04 s18 — **one figure only**, so treat the colour choice as that figure's, not a rule.
- **Clock waveforms** come two ways, both consistent within their purpose: a **train of narrow rectangular pulses** (mark shorter than space, no dashed rails) when the clock is the whole subject (L05 s18; L06 s7), and a **full-height square wave on its own labelled track with rails** when it sits alongside data signals (L05 s6, s8).
- **A vertical dashed grid aligned to the clock edges** may be laid over a multi-signal figure so the waveforms can be read against each other (L05 s8). **One figure.**

State diagrams

L08 s9 is the one slide in the whole course that *states* a diagram convention rather than merely demonstrating one. Everything here comes from it, plus the worked example on s18.

- **A state is a circle divided by a horizontal chord.** The **state name** goes in the upper half (bold); the **output value(s)** go in the lower half. Nothing else goes in the

[3] courses/csse2010/timing-diagrams.pdf

circle. (L08 s9, which prints the empty template `State Name / Output Value(s)` next to two filled examples, `S0/101` and `Alex/01`.)

- **State names are free-form.** The same slide sanctions both a systematic `S0`, `S1`, `S2`, `S3` scheme (used in the worked example, L08 s18) and an arbitrary word (`Alex`). Outputs are written as a bare bit string with no separators or subscripts: `101`, `01`, `00`, `11`.
- **A transition is a plain single-headed arrow**, labelled with a **Boolean logic expression written above the arrow**. "If expression is true, state transition is made." (L08 s9.)
- **An unlabelled arrow means "true" — always transition.** Stated in words on L08 s9.
- **Special case, exactly one input:** the label degenerates to a bare `0` or `1`. L08 s9 boxes this case in red and annotates it "Special case for only one input".
- **Transition-label algebra is the ordinary Boolean notation** of L02 s12: overbar for NOT, `.` or juxtaposition for AND, `+` for OR. L08 s9's own examples are `Z`, `RĒ SĒT`, and `A+B.C̄`.
- **Completeness:** "Each possible combination of inputs should be addressed exactly once for each state — i.e. transition arrows from each state must encompass all possibilities (exactly once)." (L08 s14.) This is a correctness rule, not a drawing rule, but it constrains what you draw.
- **Layout:** the one worked diagram (L08 s18) puts the states in a **left-to-right row** with forward transitions between neighbours and the wrap-around transition drawn as a **long arc below the row**. Single instance — the row layout is an example, not established as required.

State tables

Two forms, both defined on their own slide, both with an empty template printed so the column structure is unambiguous.

One-dimensional (L08 s15) — one row per (current state, input) combination, annotated "Every combination of state and inputs":

Current State	Input U	Next State	Outputs	
			Q1	Q0

Two-dimensional (L08 s16) — one row per state ("One row per state"), with the next-state column split by input combination ("Every combination of inputs"):

Current State	Next State		Outputs	
	$\bar{U}$	U	Q1	Q0

Structural conventions common to both: headers bold and centred; a **heavy rule under the header block** and **heavy vertical rules between column groups**, thin rules inside a group; the individual output bits are sub-columns under a single spanning `Outputs` header; the input name appears in the header, not just its values.

For a counter with **no external input**, the same table is written with the next-state columns named after the flip-flop D inputs rather than the state: `Current State (Q , Q )` | `Next State (D , D )` (L07 s11). L08 s21 lists the design pipeline these tables sit in: state definition → state diagram → state table (1D or 2D) → state encoding → Boolean expressions → logic diagram → Logisim → hardware.

The characteristic table of a flip-flop uses different headers again: `D` | `Q(t+1)`, with the right-hand column glossed "What the output will be on the next clock edge" (L05 s7).

Block and datapath diagrams

- **Plain rectangles.** No rounded corners, no drop shadows, no fill in the line-art figures. The block name is centred inside, on one or two lines. (L04 s8; L05 s17; L06 s6; L07 s16; L08 s3, s7; L09 s15, s16.)
- **Flow left to right; feedback routed around the outside**, usually below the blocks, returning to the left edge of the first block (L05 s17; L06 s6; L07 s16; L08 s3, s7). *Consistency:* every sequential-circuit block diagram in the course, across four decks.
- **Every connection gets an arrowhead at its destination.**
- **The clock enters on its own labelled stub**, not bundled with the data inputs — `Clock pulses` entering from below-left in the Mano figures (L05 s17; L06 s6; L08 s3), `Clock` entering from above in the UQ redraw (L08 s7).
- **Multi-bit paths are drawn heavy, single-bit control thin.** L09 s15/s16 draw the A, B and Dest buses as thick black arrows against thin control lines; the caption "Buses (A,B,C)" points at them. L09 s5 and s9 draw the ALU's `A`, `B`, `Output` and `Function/Control` as heavy arrows with the width lettered beside them (`n`, `n`, `n`, `m`).
- **Grouped signals are collected with a curly brace and one text label** rather than labelled individually: `Data input A`, `Data output` (L04 s7); `Control registers`, `To and from memory` (L09 s15); `Before` / `Today` (L12 s2); the IO-board input/output groups (*Device Pinouts* p. 2).
- **Anonymous repetition uses a vertical ellipsis** between the first and last drawn line, with the count given in words beside the group: "n inputs", "m outputs" (L04 s8).
- **Bus width** is marked two ways — see Inconsistencies.

Register and memory layout diagrams

Bit fields

A single row of equal-width boxes, MSB on the left, LSB on the right. Bit positions are numbered **above** the boxes, descending left to right and ending at `0`. Values or field names go inside the boxes.

Evidence: L01 s12 (a byte, `0 1 0 1 0 1 1 1`), s13 (positions `9 8 7 … 0` above, powers `2 … 2` and decimal weights `512 … 1` inside), s16 (`MSB`/`LSB` called out by red arrows to the extreme boxes), s17; L10 s15. *Consistency:* every bit-field figure. L01 s16 states the underlying rule: "the MSB is worth 2^(n-1)".

The AVR-datasheet register diagram

For a named hardware register the three-row datasheet form is used: **bit numbers `7 … 0` above the row of boxes, flag names inside the boxes (bold), an `R/W` access row underneath, and an initial-value row (`0`) under that.** (L10 s15, the AVR status register `I T H S V N Z C`.) One figure in the sampled decks, but it is verbatim datasheet form and is the obvious template for any other AVR register.

Memory maps

L11 s14, the one memory map with an explicit source (`[From ATmega324A datasheet]`):

- **Vertical stack of rectangles, lowest address at the top, highest at the bottom.**
- **Hex addresses written outside, to the right**, aligned with the region boundaries — as a range for a fully-specified region (`0x0000 - 0x001F`) or as the single start address of an open region (`0x0100`), with the final address at the bottom (`0x08FF`).
- **Hex is written `0x` + a fixed number of digits**, four for the address space (`0x0000`), two for the register file (`0x00 … 0x1F`, L11 s22).
- **Region names inside the boxes**, with the size in parentheses where useful (`Internal SRAM (1024/2048/4096/16384 x 8)`).
- **Grey fill for the large or coarsely-described regions, white for the enumerated ones.**
- **A diagonal break across the block** shows the drawing is not to scale (between Application Flash Section and Boot Flash Section).
- Column heading above each stack (`Data`, `Program Memory`).

The generic memory-organisation figure (L11 s5) agrees on direction — addresses `0 1 2 3 …` down the **left** side, increasing downward — and adds a double-headed arrow across the top labelled `Cell width`. **The alignment figure disagrees; see Inconsistencies.**

The register file

L11 s22: vertical stack, `R0` at the top through `R31` at the bottom, hex address to the right of each row, an ellipsis row `...` for the elided middle, a **heavy horizontal rule at the R15/R16 boundary** marking the split that matters for `ldi`, and free text further right for the X/Y/Z aliases (`X-register Low Byte`, etc.).

Shift and rotate

L12 s10, s11: **the printed figure is only a row of empty boxes**. The direction arrows inside the boxes, the `0` entering the vacated end, the separate small box labelled `C` for the carry, and the wrap-around arc for a rotate are all drawn in the lecturer's ink. Record this as the idiom the course *uses*, but note that **nothing about it is printed** — see Not established by the source.

AVR architecture figures

- **The AVR CPU architecture diagram (L10 s10) and the ATmega324A chip block diagram (L10 s11) are datasheet reproductions**, introduced only by the prose line "Some datasheet extracts follow" on L10 s9. Neither carries a caption. The only in-figure datasheet attribution anywhere is on the memory maps, L11 s14.
- **CPU diagram style:** black line art, no fill. The **8-bit data bus is a heavy vertical black bar** running the full height, labelled `Data Bus 8-bit` at the top. Blocks attach to the bus with **double-headed arrows**; instruction and control flow use **thin single-headed arrows**. The ALU inside is the notched trapezoid. Path labels that run vertically are set as **rotated text** (`Direct Addressing`, `Indirect Addressing`). Block names are two or three centred lines (`Flash Program Memory`, `32 x 8 General Purpose Registers`, `Instruction Decoder`, `Data SRAM`).
- **Chip-level diagram style (L10 s11):** a **dashed rectangle is the package boundary**, with pin and port names outside it (`VCC`, `GND`, `RESET`, `XTAL1`, `XTAL2`, `PA7..0`, `PB7..0`, `TOSC2/PC7`). A **heavy solid outline inside is the internal bus**, drawn with an arrowhead to show its direction. Peripherals are plain rectangles named for the function, with the width in parentheses for ports (`PORT A (8)`). Pin ranges are written `PA7..0`, high first.
- **The generic CPU figure the AVR specialises** (L09 s3, s4; L10 s3) is a heavy outer rectangle labelled `CPU` containing three filled boxes — `Control Unit`, `Arithmetic Logic Unit (ALU)`, `Registers` — with `Primary Memory` as a separate box outside it. This one *is* filled with colour (green, cyan, lilac), used only to separate adjacent blocks. See avr-architecture[4] and cpu-organisation[5].

[4] courses/csse2010/avr-architecture.pdf
[5] courses/csse2010/cpu-organisation.pdf

Truth tables and function tables

- **Inputs first (left), outputs last (right), separated by a heavier vertical rule.** (L02 s6–s10; L03 s11, s14; L04 s11, s13; L05 s10.) *Consistency:* every truth table sampled.
- **Rows in ascending binary counting order of the input columns, starting from all-zeros**, all 2 of them. (Every table sampled.) L02 s13 states the underlying instruction: "Lists output for all 2 combinations of inputs — Best to list inputs in a systematic way".
- **Column headers are the bare signal names**, bold. No units, no types.
- **Function tables for MSI blocks** list only the selector combinations, and the output column holds the **name of the selected signal**, not a bit: `S1 S0 | F` with entries `D0`, `D1`, `D2`, `D3` (L04 s9, s11). The header is called out as `Function table` (L04 s9) or `Function Table` (L04 s11).
- **A highlighted cell or row** uses a pale fill — pale orange for the active decoder output (L04 s13), pale grey for the changing bits of a counting sequence (L07 s9).
- **Visual styling of tables is not fixed** — see Inconsistencies. Only the column semantics and row ordering are stable.
- **Multi-column comparison tables** (Intel-64 vs AVR data-type support, L12 s18) use a check mark for "supported" and an empty cell for "not supported" — no cross, no "N/A".

Boolean and textual notation

L02 s12 is titled "Boolean Algebra conventions" and states them outright — the only place in the course where notation is legislated rather than demonstrated:

- **Inversion: overline.** `NOT(A) =` $\bar{A}$, "pronounced as A bar".
- **AND: a dot, or implied by adjacency.** `AND(A,B) = AB = A.B`.
- **OR: a plus sign.** `OR(A,B,C) = A+B+C`.
- **XOR:** `. XOR(A,B) = A B =` $\bar{A}B + A\bar{B}$.
- The overline extends over exactly the sub-expression negated: `NAND(A,B,C) = A B C` as one bar over `ABC`; `NOR(A,B) = A + B` as one bar over `A + B`. Nested negation stacks bars (L02 s22: `A  + B`).

Held throughout the semester (L02 s16–s23; L03 s14; L04 s10; L07 s11, s14; L08 s9; L09 s11).

Beware the operator clash the course flags itself: in an arithmetic context `+` means addition, not OR — L04 s5 says " '+' in this case means addition (not OR)".

Indexed signals take subscripts: `Q` , `Q` , `D` , `D` , `S` , `S` , `A` …`A` , `C` , `F` , `F` (L03 s16; L04 s9; L07 s10–s12; L09 s6, s10). Not universally — see Inconsistencies.

Numbers

- **Radix by subscript:** `26` , `101` , `101` , `8  - 9` (L01 s19, s20; L09 s22).
- **Radix by prefix:** `0x` (C and AVR), `$` (AVR assembly), `0b` (binary), leading `0` (octal). L01 s20 lists every alternative in a table and marks these ones "We'll mostly use these", ruling out the trailing-`h`, trailing-`q`, trailing-`b`, `@` and `%` forms.
- **Binary literals are grouped in fours with a space:** `0000 1101 0111 0111`, `00001010`, `1110 KKKK dddd KKKK` (L01 s18; L03 s8, s9; L10 s4, s16–s18). Short literals under about 8 bits are often left ungrouped (`00001111`, L11 s16).

AVR assembly and machine code

- **Machine-code layouts write fixed bits as `0/1` and each variable field as a repeated letter, one letter per bit** — `0000 11rd dddd rrrr` (ADD), `1001 010d dddd 0000` (COM), `1110 KKKK dddd KKKK` (LDI). `d` = destination register, `r` = source register, `K` = constant. Note the case split: register-field letters lowercase, the constant uppercase. (L10 s16, s17, s18.)
- **Register-transfer notation uses a left arrow:** `r3 ← r3 + r4`, `r22 ← r22 + 1` (L10 s8, s12, s13).
- **Multi-byte quantities are written as a `high:low` register pair:** `r5:r4`, `r11:r10`, `r13:r14`, `R31:R30` (L11 s22; L12 s12, s20). High byte first, always.
- **Code is set in a monospace font**, and concrete examples in the body text are coloured red (`mov r3, r14`, `add r5,r6`, `ldi r16,73` — L10 s12–s14).
- **Mnemonic case is not settled** — see Inconsistencies.

Colour and annotation

Colour carries no defined meaning anywhere in this course. No slide and none of the three guides ever says what a colour signifies. What can be observed:

- **Red is the usual colour for a callout pointing into a figure** — an arrow plus a short label, sitting outside the drawing (L01 s13 `Bit position`, `Value (Powers of 2)`; L01 s16 `MSB`/`LSB`; L02 s14 the four connection callouts; L04 s4 `Full-adders`; L09 s5 `Function/Control`; L09 s15 `Buses (A,B,C)`; L11 s5 `Address/Cell width`).
- **Blue is the usual colour for an explanatory label attached to a figure element** (L04 s15 `Logic 1`/`Logic 0`, s18 `Propagation delay`; L05 s15, s17 `Logic Gates`; L09 s10 `Carry ripples through from right to left`).
- **But the two are swapped freely.** L04 s15 uses red for `Change in input causes output to change`; L08 s15/s16 use red for both the brace and its explanation; L09 s20 uses red for the `Status` callout with a green-filled fact box beside it; L09 s15 uses purple.

- **Fills are decorative,** used only to separate adjacent blocks: yellow for the levels-of-abstraction stack (L01 s9; L02 s5), green/cyan/lilac for the three CPU parts (L09 s3, s4; L10 s3), pale blue/green/lilac for the fetch-decode-execute fact boxes (L09 s13).
- **Table highlighting** is a pale fill on the cell or row of interest (L04 s13; L07 s9).

Handwritten ink on the slides is the lecturer's live annotation, not part of the published figure. It is present on most technical slides, in red, blue, purple, green and orange, and the colour there tracks nothing but which pen was in hand — L11 s13 uses three ink colours in one worked example purely to separate three sub-workings. **Nothing that exists only in ink should be treated as an established figure convention**; where a convention below is ink-only it is flagged as such and repeated under *Not established*.

Inconsistencies

Places where the course does the same thing more than one way. **None of these is resolved here — a decision is needed before a new diagram commits to one.**

1. **The ALU has two symbol shapes, in one deck.** A **plain rectangle** with `ALU` centred and heavy arrows on four sides (L09 s5, s9), and a **notched trapezoid** with operands in the top and result out the bottom (L09 s15, s16, s20; also the AVR datasheet figure, L10 s10). Both are the ALU of the same machine, ten slides apart. *Decision needed.*

2. **The flip-flop clock input has four labels.** `CLK` (L05 s5, s6; L06 s4; L07 s10, s11, s12; *Device Pinouts* p. 2 on the 74HCT175), `CK` (L05 s14 — the very slide titled "D Latches and Flip-flops — Symbols Used"), italic serif `C` (L06 s8, s9 — the Mano figures), and lowercase italic `clk` as the net name on the wire (L06 s10, s11; L07 s15). *Decision needed.*

3. **The asynchronous inputs have three naming schemes.** `SET` / `CLR` (*Device Pinouts* p. 1, 74HCT74 schematic symbol; L07 s3, s12), `PRE̅` (preset) / `CLR` (L05 s13), and `R` with a bubble (Mano's register, L06 s8). *Device Pinouts* p. 2 gives the 74HCT175 only a `CLR`. *Decision needed.*

4. **"Current state" vs "present state", and "next state" vs "Q(t+1)".** Printed slides say `Current State` (L07 s11; L08 s15, s16); the lecturer's worked tables say `Present State` (L07 s8, s14 — ink). The flip-flop characteristic table calls the next-state column `Q(t+1)` (L05 s7) where the state table calls it `Next State` (L07 s11; L08 s15). *Decision needed.*

5. **Three textbook figure styles coexist, none declared canonical. Mano**, *Digital Design* 3rd ed. — black-and-white line art, thin rules, italic serif signal names with subscripts (`B` , `A` , `C` , `S` , `SI`, `SO`, `CLK`), plain square blocks with italic `FA` inside (L03 s16; L04 s4, s10; L05 s17; L06 s6, s8, s9; L08 s3). **Floyd**, *Digital Fundamentals* — pale-blue and beige filled blocks, magenta arrows and magenta text labels, serif `(a) Serial`

`in/shift right/serial out` sub-captions (L07 s4, s5, s7). **Tanenbaum**, *Structured Computer Organization* — dense line art with dashed boxes grouping sub-circuits (L09 s8). And L01 s8 says there is no textbook. *Decision needed:* which family a redrawn figure should imitate.

6. **The same block diagram exists in two drawing styles.** The sequential-circuit block diagram appears as the **Mano original** — thin rules, serif lowercase labels `Combinational circuit` / `Flip-flops` / `Clock pulses` (L05 s17; L06 s6; L07 s16; L08 s3) — and as a **UQ redraw** in bold sans-serif with heavy borders (L08 s7, the Moore/Mealy slide). Same content, same course, different look. *Decision needed.*

7. **Memory addresses increase downward in some figures and upward in others.** Downward: the generic cell diagram (L11 s5, addresses `0 1 2 3 …` down the left), the AVR address-space maps (L11 s14, `0x0000` at the top), the register file (L11 s22, `R0` at the top), the endianness examples (L11 s16, s17). Upward: the alignment figure (L11 s10), where the row addresses `0 8 16 24` run bottom to top and the byte numbers within a row run high-to-low left to right (`15 14 13 12 11 10 9 8`). *Decision needed.*

8. **Hex is prefixed `0x` in some places and `$` in others — sometimes on the same slide.** `0x`: L10 s15 area, L11 s14 (`0x0000`), L11 s22 (`0x1F`), L11 s27 ("0 to 0x3F"), L12 s15 (`andi r17, 0xBA`). `$`: L11 s26 ("$20 to $FF"), L12 s15 (`lds r15, $1234`). L01 s20 blesses both. *Decision needed.*

9. **Assembly mnemonic case in the generic form.** Uppercase: `MOV rd, rr`, `ADD rd, rr`, `AND rd, rr`, `LDI rd, K` (L10 s12, s13, s14; L11 s25; L12 s4, s9). Lowercase bold-italic: `in rd, P`, `out P, rr` (L11 s27; L12 s5), and lowercase throughout the addressing-modes slide (`andi`, `and`, `lds`, `ld` — L12 s15, s16). Concrete examples are lowercase in print (`mov r3, r14`) but uppercase in the lecturer's ink (`ADD r10, r4`, L12 s20). *Decision needed.*

10. **Register names: `R0`–`R31` or `r0`–`r31`.** Uppercase in the register-file figure (L11 s22) and the AVR architecture block (L10 s10, `32 x 8 General Purpose Registers`); lowercase everywhere in prose and code (L10 s12–s14; L11 s27; L12 s4–s6). *Decision needed.*

11. **Truth-table visual styling has three treatments.** Dark-red header row with pink zebra striping (L02 s9, s10; L03 s11; L04 s11); plain black-on-white with a bold header and a heavy input/output divider (L02 s6, s7; L03 s14; L04 s13; L05 s10); light-grey header with grey cell highlighting (L07 s9). Column semantics and row order are stable across all three; only the appearance differs. *Decision needed for a house table style.*

12. **Bus width is marked two ways.** A **short slash across the line with the width beside it** (*Device Pinouts* p. 2, the `4`-marked buses of the 74HCT157 and 74HCT283 logic-diagram symbols). Or the **width simply lettered next to a plain arrow, no slash** (L09 s5, s9 — `n`, `n`, `m`, `n`). *Decision needed.*

13. **The adder's function glyph.** A large `+` in the trapezoid body (L04 s7; *Device Pinouts* p. 2). Or `Σ` on the block's output pin with `A`, `B`, `C_out` on the others (L07 s7, the Floyd figure). *Decision needed.*

14. **Indexed signals are subscripted inconsistently.** Subscripted `Q` , `Q` , `D` , `D` in the counter state table and its circuit (L07 s10, s11, s12). Unsubscripted `Q1`, `Q0` in the state-table templates (L08 s15, s16) and `S0`, `D0` in the mux function table (L04 s9). *Decision needed.*

Not established by the source

Things a CSSE2010 diagram may well need that the materials simply do not settle. **These are where invention risk lives.**

- **Karnaugh maps: entirely absent.** No K-map appears on any of the 221 deck pages sampled, nor in any of the three guides. The course's simplification path runs truth table → sum of products → Boolean identities → algebraic manipulation (L02 s16–s21), and De Morgan is done algebraically (L02 s21, s22). If a K-map is needed, **everything about it is invention**: cell ordering and whether the axes are Gray-coded, which variable goes on which axis, how variable labels are written along the edges, the style of the grouping loops, and the don't-care mark.
- **The initial/reset state of a state diagram.** L08 s9 defines only the state circle and the transition arrow. The one worked diagram (s18) has no incoming arrow from nowhere, no double circle, no `reset` label. **How to mark the start state is unspecified.**
- **Mealy outputs on a state diagram.** L08 s6, s7 define both machine types, then s7 says "We'll stick to Moore machines in this course". The notation slide therefore supplies only the two-half circle, which is a Moore output. **There is no notation for an output written on a transition** (no `input/output` slash form anywhere).
- **Self-loop transitions.** None is drawn anywhere sampled, although the completeness rule (L08 s14) guarantees most machines need them. Loop shape, side of the circle, and label placement are all unspecified.
- **The don't-care symbol.** `X` is used for don't-care next-state entries at L07 s14 and for "byte not part of this word" at L11 s10 — **both in the lecturer's ink, never in print.** No printed definition of the symbol exists.
- **The shift/rotate register idiom.** L12 s10, s11 print only a row of empty boxes; the direction arrows, the `0` entering the vacated end, the separate `C` carry box and the rotate wrap-arc are all ink. The idiom is clear, but it has **no printed authority**.
- **Numeric values written inside timing-diagram segments.** The `0/1` written in the middle of each waveform segment is always the lecturer's ink (L04 s15, s16, s17; L05 s6), never printed. Whether a finished diagram should carry them is unsettled.
- **Setup and hold time.** Only propagation delay, rise time and fall time are named and drawn (L04 s18). Setup/hold appear nowhere, so their marking is unspecified.
- **Waveforms for multi-bit buses.** The "hexagon" (split-and-rejoin) idiom for a bus carrying a value is never drawn. Every waveform in the course is a single-bit two-level trace.

- **Whether a multi-bit signal in a gate-level logic diagram should be thickened or slashed.** The datapath figures use heavy arrows (L09 s15, s16) and the guide's MSI symbols use slashes (*Device Pinouts* p. 2), but gate-level diagrams use exactly one line weight and no slashes. Nothing bridges the two.
- **Tri-state, open-collector and bidirectional signal marking.** Not drawn anywhere. The AVR datasheet figures use double-headed arrows for bidirectional bus attachment (L10 s10, s11), but that idiom is inherited, not taught.
- **Symbols for gates with three or more inputs.** L02 s9, s10 ask for 3-input NAND and XOR truth tables but never draw a 3-input symbol, and the parts list contains only 2-input gates plus the hex inverter (*Device Pinouts* p. 1; see device-pinouts[6]). How to extend the body and space the input leads is unspecified.
- **Figure numbering and captioning.** The decks caption almost nothing. The only figure caption in any deck is Mano's own, retained inside a reproduced image: `Fig. 6-3 4-Bit Shift Register` (L06 s9). Source credits sit as a bracketed line under the figure. **There is no course figure-numbering scheme**, so any new note must choose its own.
- **Stroke weights, arrowhead geometry, fonts and sizes.** No slide or guide specifies any of these numerically.
- **What colour means.** Nothing in any slide or guide assigns meaning to a colour. The near-patterns recorded above are observations, not rules — and they are broken more than once. **Do not attach semantic meaning to colour in a new diagram** unless you define it in that diagram's own caption.

See also

- circuit-schematics[7] — the schematic rules this note deliberately does not repeat: device IDs, gate letters, pin numbers, junction dots, power-supply blocks, the seven common errors
- device-pinouts[8] — 74-series pinouts, the logic-diagram vs circuit-schematic symbol columns, the IO board
- logic-gates[9] — the seven gate types and their truth tables
- timing-diagrams[10] — what a timing diagram means, as opposed to how it is drawn
- state-machines[11] — state diagrams and state tables as content
- boolean-algebra[12] — the identities the transition-label algebra uses
- avr-architecture[13] — the AVR figures described here, read for content

[6] courses/csse2010/device-pinouts.pdf
[7] courses/csse2010/circuit-schematics.pdf
[8] courses/csse2010/device-pinouts.pdf
[9] courses/csse2010/logic-gates.pdf
[10] courses/csse2010/timing-diagrams.pdf
[11] courses/csse2010/state-machines.pdf
[12] courses/csse2010/boolean-algebra.pdf
[13] courses/csse2010/avr-architecture.pdf