Xilinx 7 Series FPGA CLB

- ASMBL Architecture enabled domain specific platform FPGAs
  - Helps with scalability and power because all pins are between the columns
    - Improves scalability because if the chip grows, i.e. transistors increase then it doesn’t have to do quadratic scaling
    - Improves power because now all the pins are close to each other especially the power pins.
- Storage
  - Flip Flops, level sensitive latches can be used and there are eight storage elements per slice
- Distributed RAM
  - The function generators LUTS can be implemented as a synchronous RAM.
  - Synchronous write resources. A S.R. can be done with a flip flop on same slice
  - The distributed RAM performance is proved by lowering the delay into the clock to out value of the flip flop.
  - LUT RAM is faster than BRAM and is less wasteful for small memories. If you need async reads you use DR. BR is completely sync for read/writes
  - Makes use of some LUTS of logic fabric as memory instead of implementing them for logic good for small blocks (fine grain) i.e. 3 input LUT can provide 8-1-bit locations.
  - Advantage is that it’s also all over the chip, so the place and route can put it near the logic
  - Basically: Read -> async, write requires one clock edge, can implemented a 64 x1 bit ROM up to 256 bit. 64bit x1 -> 1 lut and 256 = 4 LUTS.
- Shift Registers
  - Can be used for delay or latency compensation or synchronous FIFO and content addressable memory
  - Can do writes, synchronous and dynamic reads.
  - Dynamic reads useful when doing 13 bit shift registers to set address to 13 bit
- Multiplexers
  - Each LUT can be configured into a 4:1 mux
  - there are four 4:1 muxes per slice and 8:1 muxes uses two LUTs which is two 8:1 MUXes per slice
  - Wide multiplexers are implemented in one level of logic
- Carry Logic
  - Fast lookahead carry logic is used to perform fast arithmetic adds/subtracts in a slice. A CLB has two separate carry chains
  - The propagation delay for an adder increases linearly with the number of bits in the operand