Introduction to HDL Design

Traditionally, digital design was done with schematic entry.

In today's competitive business environment, building cost-effective products quickly is best done with a top down methodology utilizing hardware description languages (HDLs) and synthesis.

Schematic entry still used at the board level, but this will probably also change.

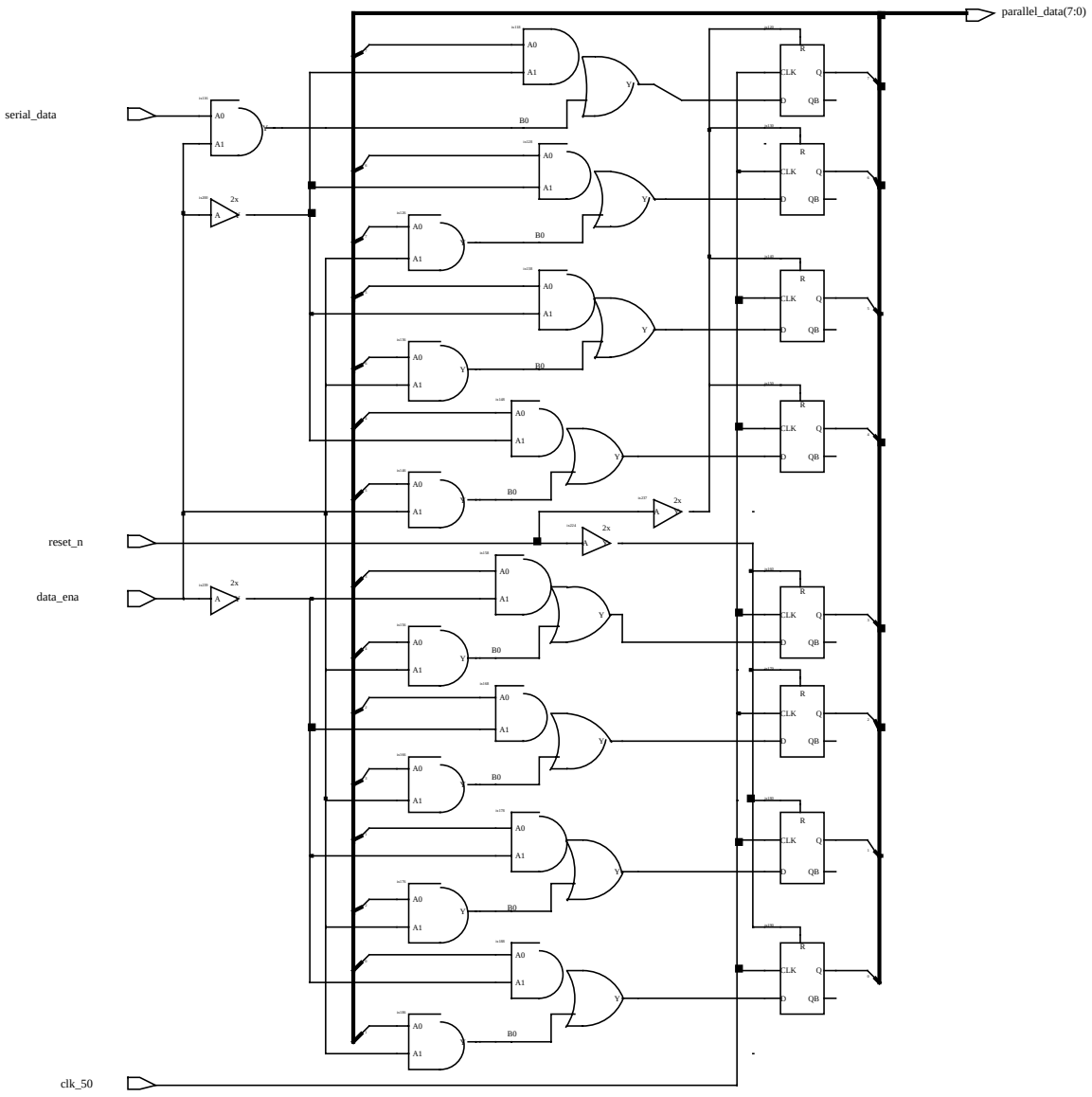
As a first example of the power of using HDLs, consider the code below. It implements a 8 bit shift register with enable. It is easially changed to about any width with a few quick key strokes. In the amount of time it takes to sneeze, it can be synthesized into the schematic on the next page.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_vector_arith.ALL;

ENTITY shift_reg IS
  PORT(
      clk_50        : IN      STD_LOGIC;  --50Mhz input clock
      reset_n       : IN      STD_LOGIC;  --reset async active low
      data_ena      : IN      STD_LOGIC;  --serial data enable
      serial_data   : IN      STD_LOGIC;  --serial data
      parallel_data : BUFFER  STD_LOGIC_VECTOR(7 DOWNTO 0)
      );
  END shift_reg;

 ARCHITECTURE struct OF shift_reg IS
 BEGIN
shift_register:
 PROCESS (clk_50, reset_n, data_ena, serial_data, parallel_data)
   BEGIN
     IF (reset_n = '0') THEN
       parallel_data <= "00000000";
     ELSIF (clk_50'EVENT AND clk_50 = '1') THEN
       IF (data_ena = '1') THEN
         parallel_data(7) <= serial_data;  --input gets input data
         FOR i IN 0 TO 6 LOOP
          parallel_data(i) <= parallel_data(i+1); --all other bits shift down
         END LOOP;
       ELSE
          parallel_data <= parallel_data;
       END IF;
     END IF;
   END PROCESS shift_register;
```

Synthsized 8-Bit Shift Register



HDLs - Motivation

Increased productivity

shorter development cycles, more features, but.....
still shorter time-to-market, 10-20K gates/day/engineer

Flexible modeling capabilities.

can represent designs of gates or systems
description can be very abstract or very structural

Design reuse is enabled.

packages, libraries, support reusable, portable code

Design changes are fast and easily done

convert a 8-bit register to 64-bits.....
four key strokes, and its done!
exploration of alternative architectures can be done quickly

Use of various design methodologies.

top-down, bottom-up, complexity hiding (abstraction)

Technology and vendor independence.

same code can be targeted to CMOS, ECL, GaAs
same code for: TI, NEC, LSI, TMSC
same code for: .5um, .35um, .25um, .18um

Enables use of logic synthesis which allows a investigation of the area and timing space.

ripple adder or CLA?, How many stages of look ahead?

HDLs can leverage software design environment tools.

vi, emacs, cvs, lint, grep, make files

Using a standard language promotes clear communication of ideas and designs.

schematic standards?... what's that... a tower of Babel.

HDLs - What are they? How do we use them?

A Hardware Description Language (HDL) is a programming language used to model the intended operation of a piece of hardware.

An HDL can facilitate:

abstract behavioral modeling

-no structural or design aspect involved

hardware structure modeling

-a hardware structure is explicitly implied

In this class we will use an HDL to describe the structure of a hardware design.

When we use an HDL to create hardware by logic synthesis, we will write code at the *Register Transfer Language (RTL)* level. At this level we are implying certain hardware structures when we understand apriori.

When programming at the RTL level, we are not describing an algorithm which some hardware will execute, we are describing a hardware structure.

Without knowing beforehand what the structure is we want to build, use of an HDL will probably produce a steaming pile (think manure) of gates which may or may not function as desired.

You must know what you want to build before you describe it in an HDL.

Knowing an HDL does not relieve you of thoroughly understanding digital design.

HDL's- VHDL or Verilog

We will use VHDL as our HDL.

VHDL

more capable in modeling abstract behavior
more difficult to learn
strongly typed
85% of FPGA designs done in VHDL

Verilog

easier and simpler to learn
weakly typed
85% of ASIC designs done with Verilog (1993)

The choice of which to use is not based solely on technical capability, but on:

personal preferences
EDA tool availability
commercial business and marketing issues

We use VHDL because

strong typing keeps students from getting into trouble
if you know VHDL, Verilog can be picked up in few weeks
If you know Verilog, learning VHDL will take several months

The Bottom line...Either language is viable. Employers don't care as long as you know one of them.

VHDL - Origins

Roots of VHDL are in the Very High Speed Integrated Circuit (VHSIC) Program launched in 1980 by the US Department of Defense (DOD).

The VHSIC program was an initiative by the DOD to extend integration levels and performance capabilities for military integrated circuits to meet or exceed those available in commercial ICs.

The project was successful in that very large, high-speed circuits were able to be fabricated successfully. However, it became clear that there was a need for a standard programming language to describe and document the function and structure of these very complex digital circuits.

Therefore, under the VHSIC program, the DOD launched another program to create a standard hardware description language. The result was the VHSIC hardware description language or VHDL.

The rest is history...

In 1983, IBM, TI and Intermetrics were awarded the contract to develop VHDL.

In 1985, VHDL V7.2 released to government.

In 1987, VHDL became IEEE Standard 1076-1987.

In 1993, VHDL restandardized to clarify and enhance the language resulting in VHDL Standard 1076-1993.

In 1993, development began on the analog extension to VHDL, (VHDL-AMS).

Extends VHDL to non-digital devices and micro electromechanical components. This includes synthesis of analog circuits.

Some Facts of Life (For ASIC designers)

The majority of costs are determined by decisions made early in the design process.

"Hurry up and make all the mistakes. Get them out of the way!"

"Typical" ASIC project: concept to first silicon about 9 months.

95% of designs work as the specification states.

60% of designs fail when integrated into the system.

The design was not the right one, but it "works".

Technology is changing so fast, the only competitive advantage is to learn faster than your competitors.

To design more "stuff" faster, your level of abstraction in design must increase.

Using HDLs will help to make digital designers successful. (and employed!)