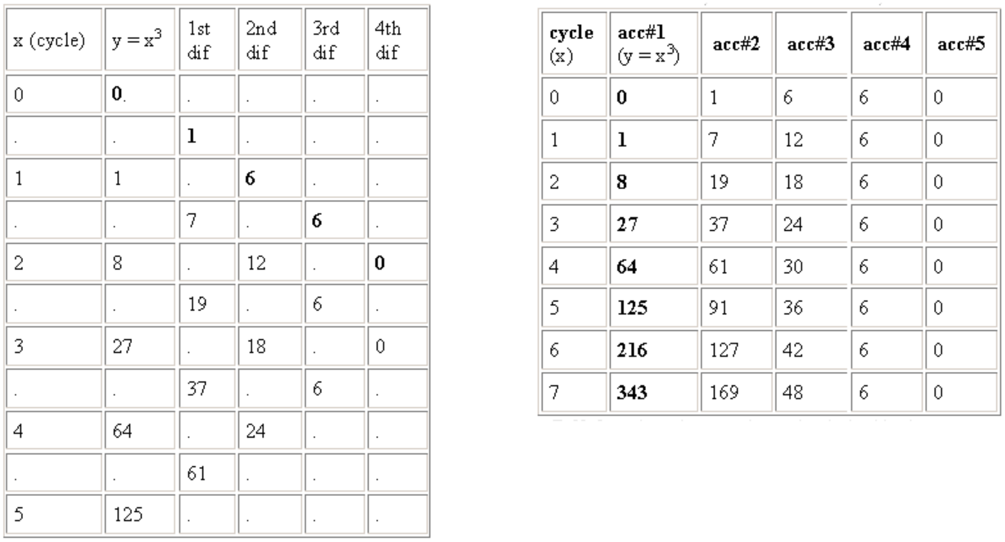
ICAR Difference Engine Number 3

Babbage was commissioned by the British government to provide accurate Log, Trig and other tables for naval navigation. His Difference Engine No. 1 (DE1) was started in 1822, however, funding later withdrawn due to increasing costs. He redesigned this machine during 1847 – 1849, producing Difference Engine No. 2 (DE2). However, Babbage only completed partial prototypes, he did not produce a fully working machine (later completed by the British Science museum).

DE2 evaluated polynomials using the method of finite differences, e.g. to repeatedly calculate the values of $y = x^3$ we would manually calculate the first five values, taking the repeated difference of these values as shown in the left table of figure 1. This produces a set of initial coefficients i.e. 0, 1, 6, 6, and 0. Using the DE2 accumulator based architecture as shown in figure 2, these coefficients can be used to regenerate increasing values of x, as shown in the right table of figure 1.



x (cycle)	$y = x^3$	1st dif	2nd dif	3rd dif	4th dif
0	**0**.	.	.	.	.
.	.	**1**	.	.	.
1	1	.	**6**	.	.
.	.	7	.	**6**	.
2	8	.	12	.	**0**
.	.	19	.	6	.
3	27	.	18	.	0
.	.	37	.	6	.
4	64	.	24	.	.
.	.	61	.	.	.
5	125	.	.	.	.

cycle (x)	**acc#1 ($y = x^3$)**	**acc#2**	**acc#3**	**acc#4**	**acc#5**
0	**0**	1	6	6	0
1	**1**	7	12	6	0
2	**8**	19	18	6	0
3	**27**	37	24	6	0
4	**64**	61	30	6	0
5	**125**	91	36	6	0
6	**216**	127	42	6	0
7	**343**	169	48	6	0

Figure : calculating the initial coefficients for $y = x^3$

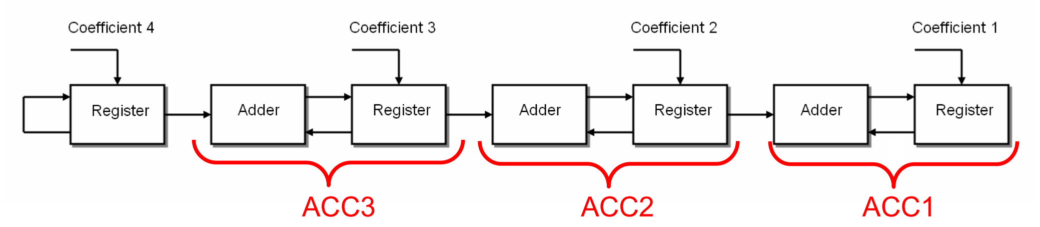


Figure 2 : DE2 architecture

For more information on this algorithm and DE2 refer to:

http://ed-thelen.org/bab/bab-intro.html

In general a computer's architecture is independent of the technology used to implement it, however, there may be practical limitations due to size, cost, power etc. Therefore, the simplified architectural block diagram for DE2 shown in figure 3 can now be implemented in silicon i.e. the new and improved, Difference engine mark 3 (DE3).

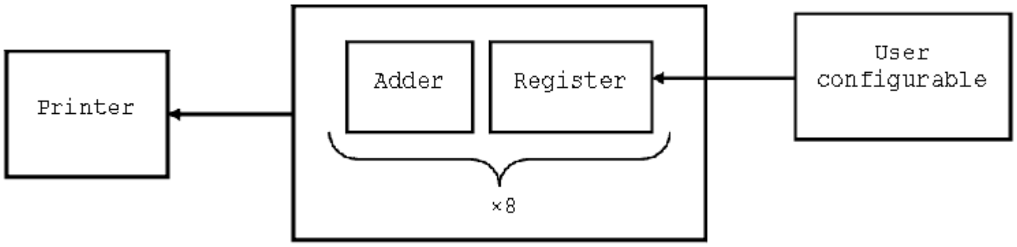
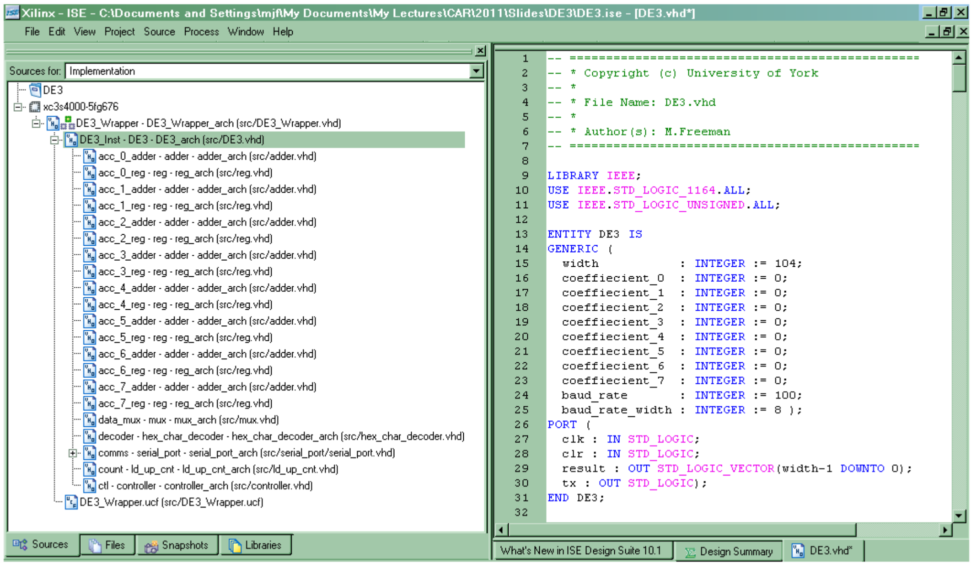


Figure 3 : Simplified block diagram

To implement this architecture's hardware a Field Programmable Gate Array (FPGA) will be used i.e. user configurable hardware. For more information on FPGA devices refer to:

http://en.wikipedia.org/wiki/Field-programmable_gate_array

To define this architecture's hardware units a textual VHDL description will be used. VHDL is a double acronym: V standing for very high speed integrated circuit (VHSIC), HDL for hardware description language. This language was commissioned by the American Department of Defence in the early 1980's as an implementation independent method of describing electronic systems, especially hardware. The DE3 hardware is defined in a number of text file as shown in figure 4.


Xilinx - ISE - C:\Documents and Settings\mjf\My Documents\My Lectures\CAR\2011\Slides\DE3\DE3.ise - [DE3.vhd*]
File
Edit
View
Project
Source
Process
Window
Help
Sources
Files
Snapshots
Libraries
What's New in ISE Design Suite 10.1
Design Summary
DE3.vhd*

Sources for: Implementation

```
DE3
xc3s4000-5fg676
  DE3_Wrapper - DE3_Wrapper_arch (src/DE3_Wrapper.vhd)
    DE3_Inst - DE3 - DE3_arch (src/DE3.vhd)
      acc_0_adder - adder - adder_arch (src/adder.vhd)
      acc_0_reg - reg - reg_arch (src/reg.vhd)
      acc_1_adder - adder - adder_arch (src/adder.vhd)
      acc_1_reg - reg - reg_arch (src/reg.vhd)
      acc_2_adder - adder - adder_arch (src/adder.vhd)
      acc_2_reg - reg - reg_arch (src/reg.vhd)
      acc_3_adder - adder - adder_arch (src/adder.vhd)
      acc_3_reg - reg - reg_arch (src/reg.vhd)
      acc_4_adder - adder - adder_arch (src/adder.vhd)
      acc_4_reg - reg - reg_arch (src/reg.vhd)
      acc_5_adder - adder - adder_arch (src/adder.vhd)
      acc_5_reg - reg - reg_arch (src/reg.vhd)
      acc_6_adder - adder - adder_arch (src/adder.vhd)
      acc_6_reg - reg - reg_arch (src/reg.vhd)
      acc_7_adder - adder - adder_arch (src/adder.vhd)
      acc_7_reg - reg - reg_arch (src/reg.vhd)
      data_mux - mux - mux_arch (src/mux.vhd)
      decoder - hex_char_decoder - hex_char_decoder_arch (src/hex_char_decoder.vhd)
      comms - serial_port - serial_port_arch (src/serial_port/serial_port.vhd)
      count - ld_up_cnt - ld_up_cnt_arch (src/ld_up_cnt.vhd)
      ctl - controller - controller_arch (src/controller.vhd)
    DE3_Wrapper.ucf (src/DE3_Wrapper.ucf)
```

```vhdl
-- ==============================================
-- * Copyright (c) University of York
-- *
-- * File Name: DE3.vhd
-- *
-- * Author(s): M.Freeman
-- ==============================================

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY DE3 IS
GENERIC (
  width           : INTEGER := 104;
  coeffiecient_0  : INTEGER := 0;
  coeffiecient_1  : INTEGER := 0;
  coeffiecient_2  : INTEGER := 0;
  coeffiecient_3  : INTEGER := 0;
  coeffiecient_4  : INTEGER := 0;
  coeffiecient_5  : INTEGER := 0;
  coeffiecient_6  : INTEGER := 0;
  coeffiecient_7  : INTEGER := 0;
  baud_rate       : INTEGER := 100;
  baud_rate_width : INTEGER := 8 );
PORT (
  clk : IN STD_LOGIC;
  clr : IN STD_LOGIC;
  result : OUT STD_LOGIC_VECTOR(width-1 DOWNTO 0);
  tx : OUT STD_LOGIC);
END DE3;

```

Figure 4 : VHDL implementation

For more information on the VHDL language refer to:

http://en.wikipedia.org/wiki/VHDL

Like the original DE2, this new implementation only performs the algorithm of the method of finite differences. Initial coefficient values are passed through generic parameters as shown in the left panel of figure 4. When initialised these text files are then synthesized into a configuration bit file that sets up the required internal FPGA connections to realise the DE3 architecture in silicon. The RTL schematic (circuit diagram) and the FPGA floor plan (layout of the hardware elements) for this architecture are shown in figure 5.

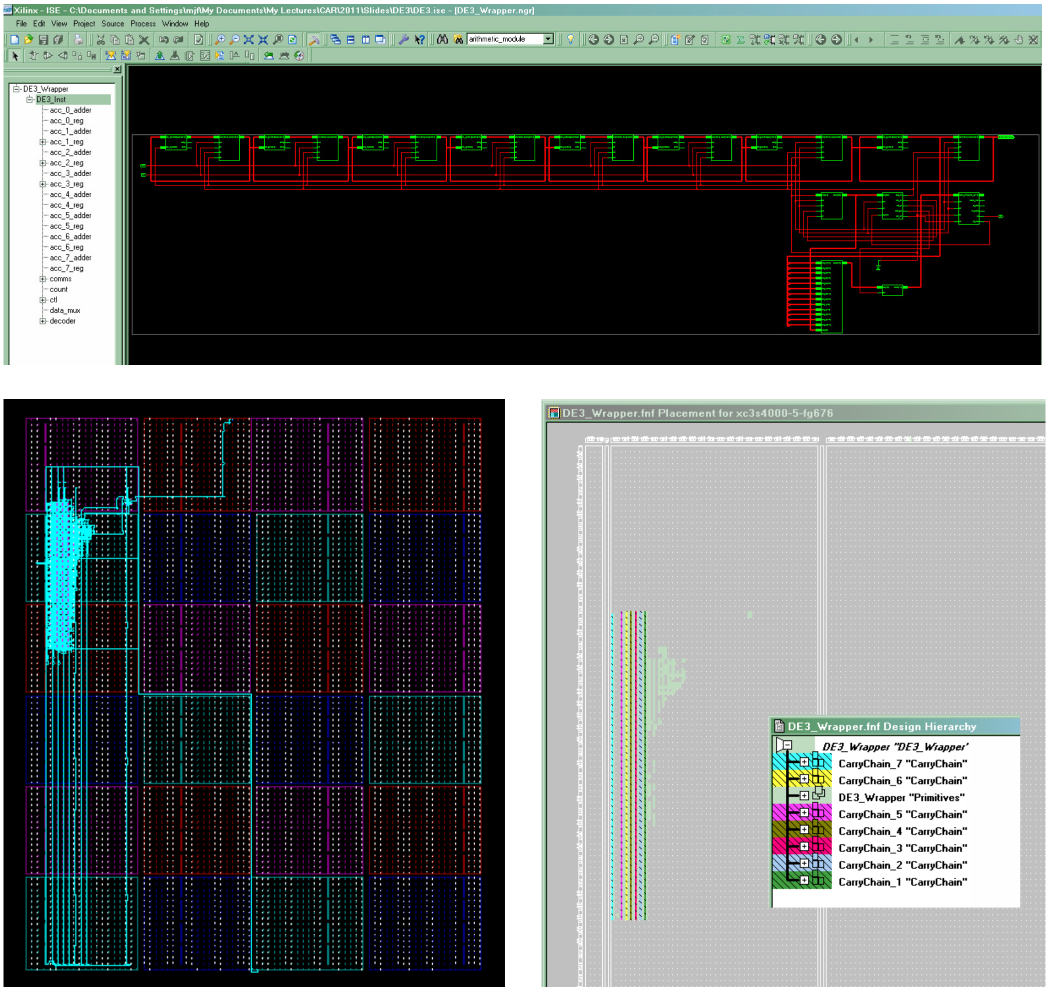
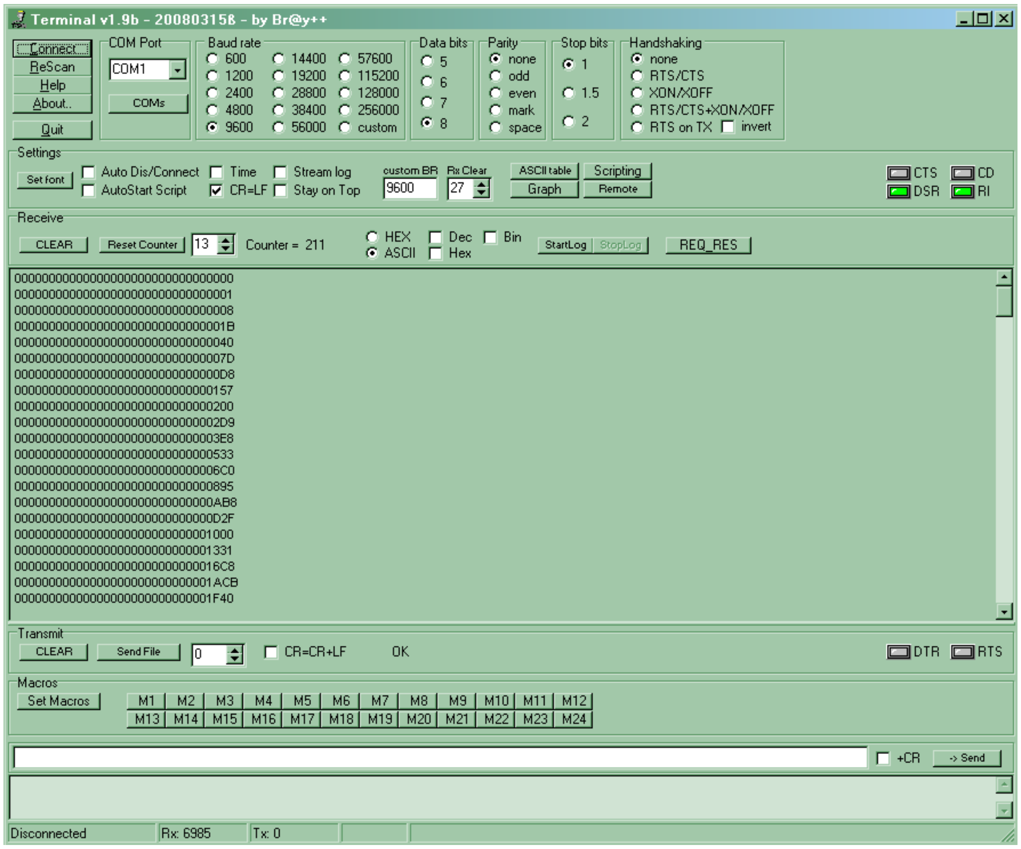

Xilinx - ISE - C:\Documents and Settings\mjf\My Documents\My Lectures\CAR\2011\Slides\DE3\DE3.ise - [DE3_Wrapper.ngr]
File Edit View Project Source Process Window Help
arithmetic_module
DE3_Wrapper
DE3_Inst
acc_0_adder
acc_0_reg
acc_1_adder
acc_1_reg
acc_2_adder
acc_2_reg
acc_3_adder
acc_3_reg
acc_4_adder
acc_4_reg
acc_5_adder
acc_5_reg
acc_6_adder
acc_6_reg
acc_7_adder
acc_7_reg
comms
count
ctl
data_mux
decoder
DE3_Wrapper.fnf Placement for xc3s4000-5-fg676
DE3_Wrapper.fnf Design Hierarchy
DE3_Wrapper "DE3_Wrapper'
CarryChain_7 "CarryChain"
CarryChain_6 "CarryChain"
DE3_Wrapper "Primitives"
CarryChain_5 "CarryChain"
CarryChain_4 "CarryChain"
CarryChain_3 "CarryChain"
CarryChain_2 "CarryChain"
CarryChain_1 "CarryChain"

Figure 5 : RTL schematic and FPGA Floor plan

Note, the eight repeated accumulator (ACC) units can be seen in the top of the schematic, also an idea of the approximate size of this implementation can be gained from the floor paln layout. Using this FPGA there is more than enough space for multiple difference engines to be implemented on the same FPGA. To display the results from the DE3's output register an additional RS232 interface has been added to this design. This converts the output register's 104bit result into a hexadecimal ASCII representation that can be transmitted to a PC. For more information on this interface refer to:

http://en.wikipedia.org/wiki/ASCII
http://en.wikipedia.org/wiki/RS-232

Using a terminal program running on a PC the data values generated by the DE3 can be displayed to the user as shown in figure 6. These data values are displayed as 32 digital hexadecimal values i.e. base 16.



COM Port: COM1
Baud rate: ● 9600
Data bits: ● 8
Parity: ● none
Stop bits: ● 1
Handshaking: ● none
Settings: ☐ Auto Dis/Connect ☐ AutoStart Script ☐ Time ☑ CR=LF ☐ Stream log ☐ Stay on Top
custom BR: 9600
Rx Clear: 27
Receive: 13 Counter = 211 ○ HEX ● ASCII ☐ Dec ☐ Hex ☐ Bin

```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000008
0000000000000000000000000000001B
00000000000000000000000000000040
0000000000000000000000000000007D
000000000000000000000000000000D8
00000000000000000000000000000157
00000000000000000000000000000200
000000000000000000000000000002D9
000000000000000000000000000003E8
00000000000000000000000000000533
000000000000000000000000000006C0
00000000000000000000000000000895
00000000000000000000000000000AB8
00000000000000000000000000000D2F
00000000000000000000000000001000
00000000000000000000000000001331
000000000000000000000000000016C8
00000000000000000000000000001ACB
00000000000000000000000000001F40
```

Transmit: 0 ☐ CR=CR+LF OK
☐ +CR

Disconnected Rx: 6985 Tx: 0

Figure 6 : DE3 results for $y = x^3$

Converting these hexadecimal values into decimal values we can see that we have regenerated the $y = x^3$ values as defined in figure 1.

$0000_{16} = 0000_{10}$
$0001_{16} = 0001_{10}$
$0008_{16} = 0008_{10}$
$001B_{16} = 0027_{10}$
$0040_{16} = 0064_{10}$
$007D_{16} = 0125_{10}$
$00D8_{16} = 0216_{10}$
$0157_{16} = 0343_{10}$
...

The complete VHDL description for DE3 can be downloaded from the module web page.