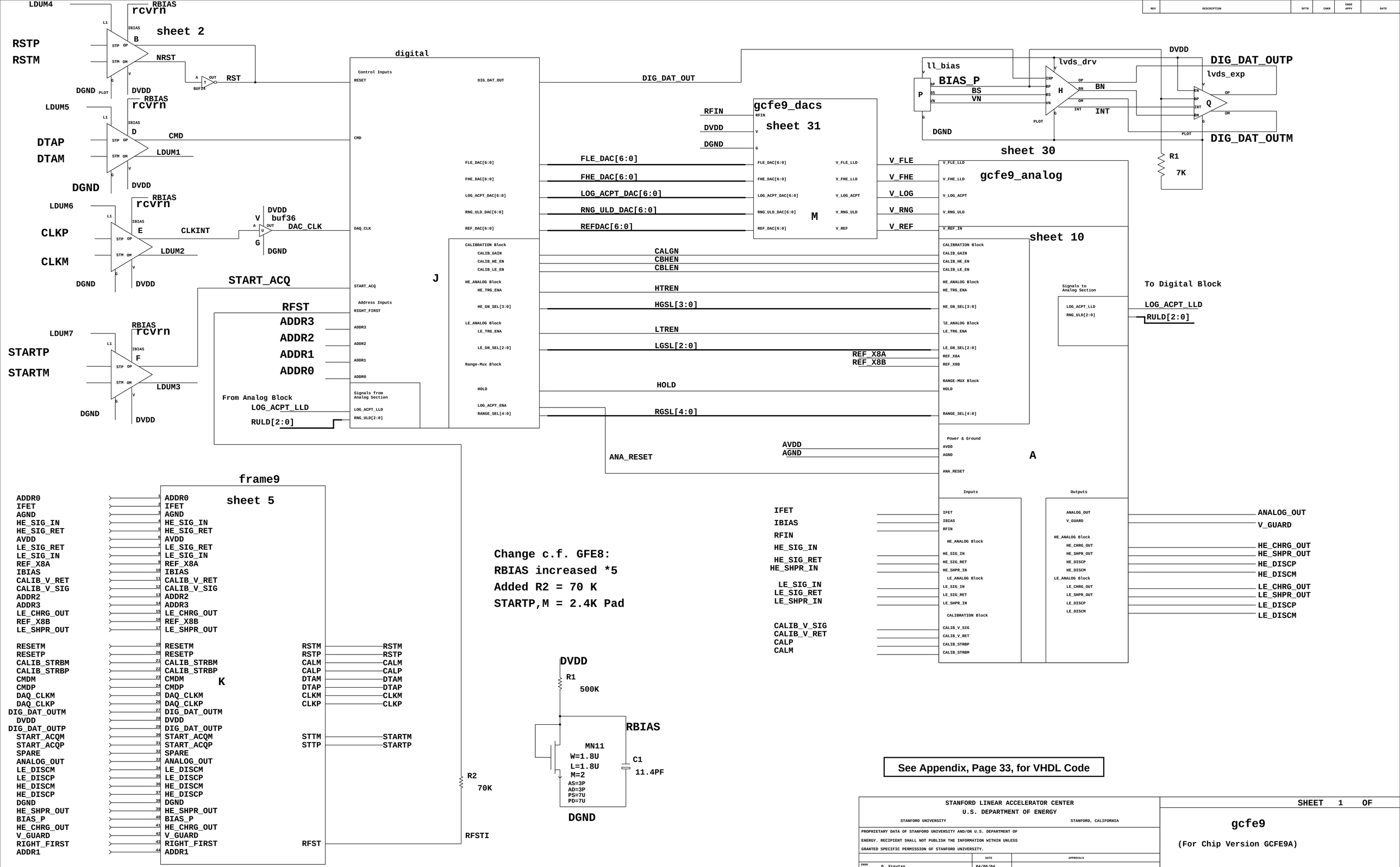
REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY — STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET 1 OF

gcfe9

(For Chip Version GCFE9A)

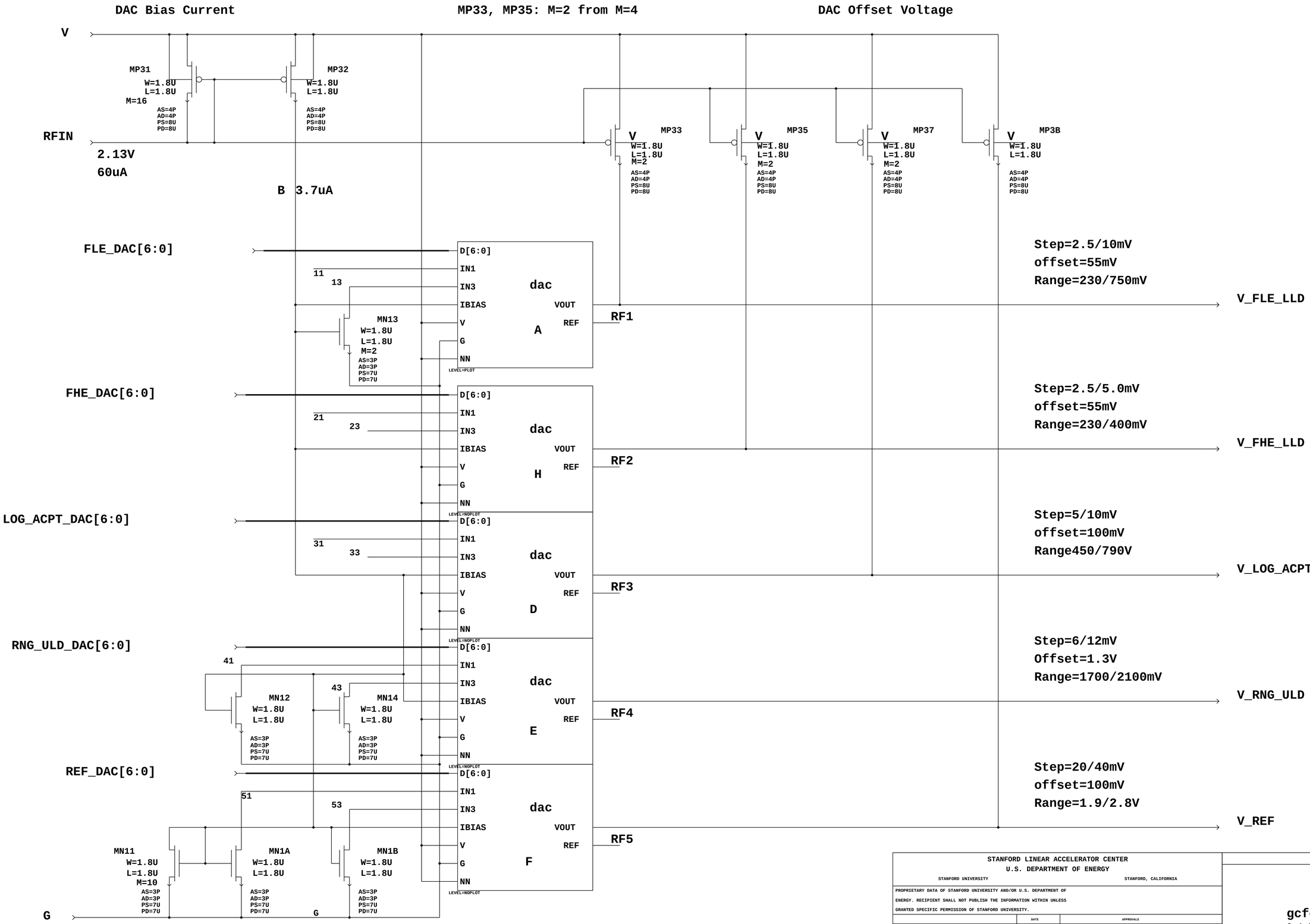
4-5-2004_16:44

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE

Change c.f. gcfe4_dacs:
MP33, MP35: M=2 from M=4



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

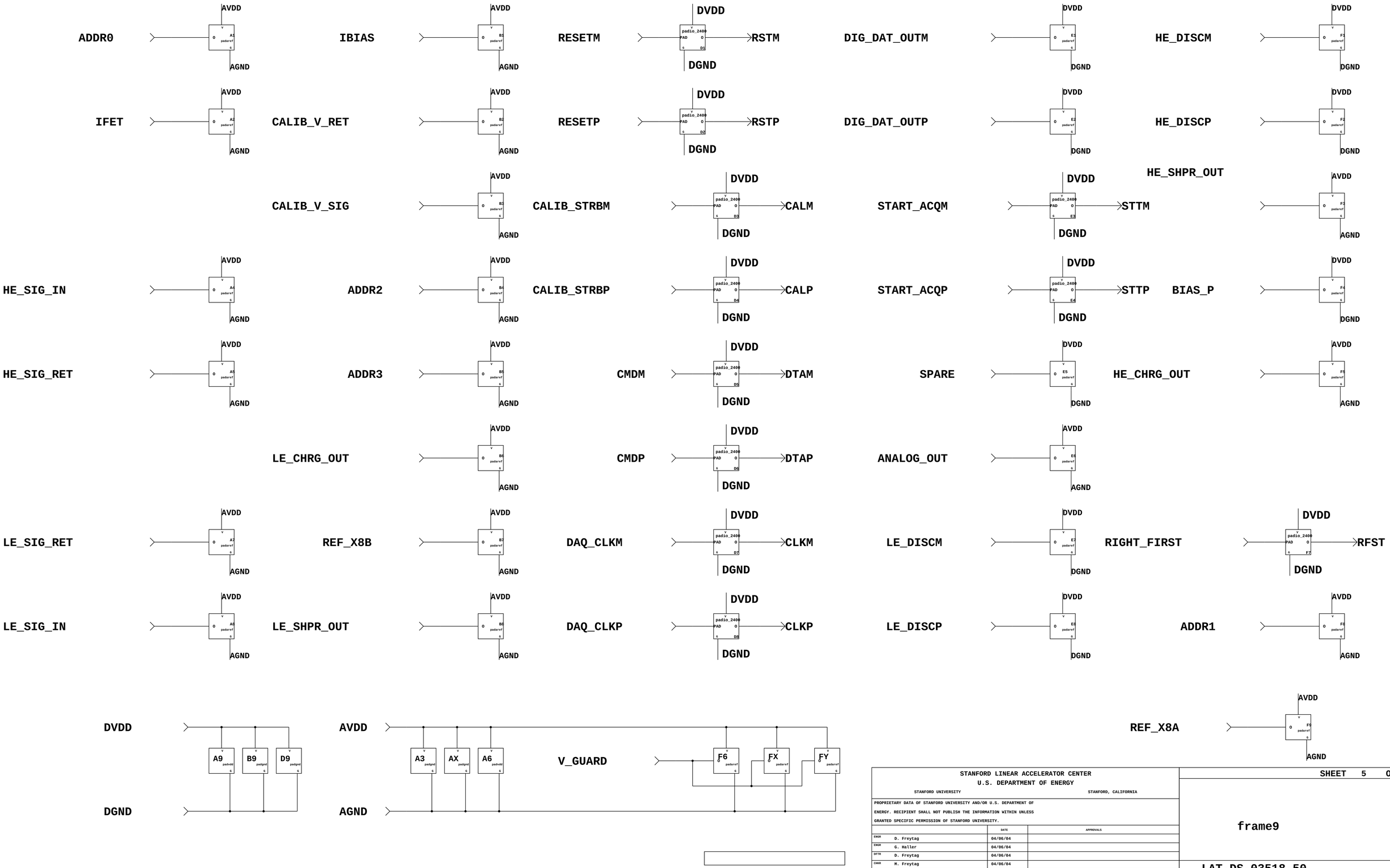
		DATE	APPROVALS
ENGR	D. Freytag	04/08/08	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET 31 OF

gcfe9_dacs
6-4-2003_14:52

LAT-DS-03518-50 D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET 5 OF

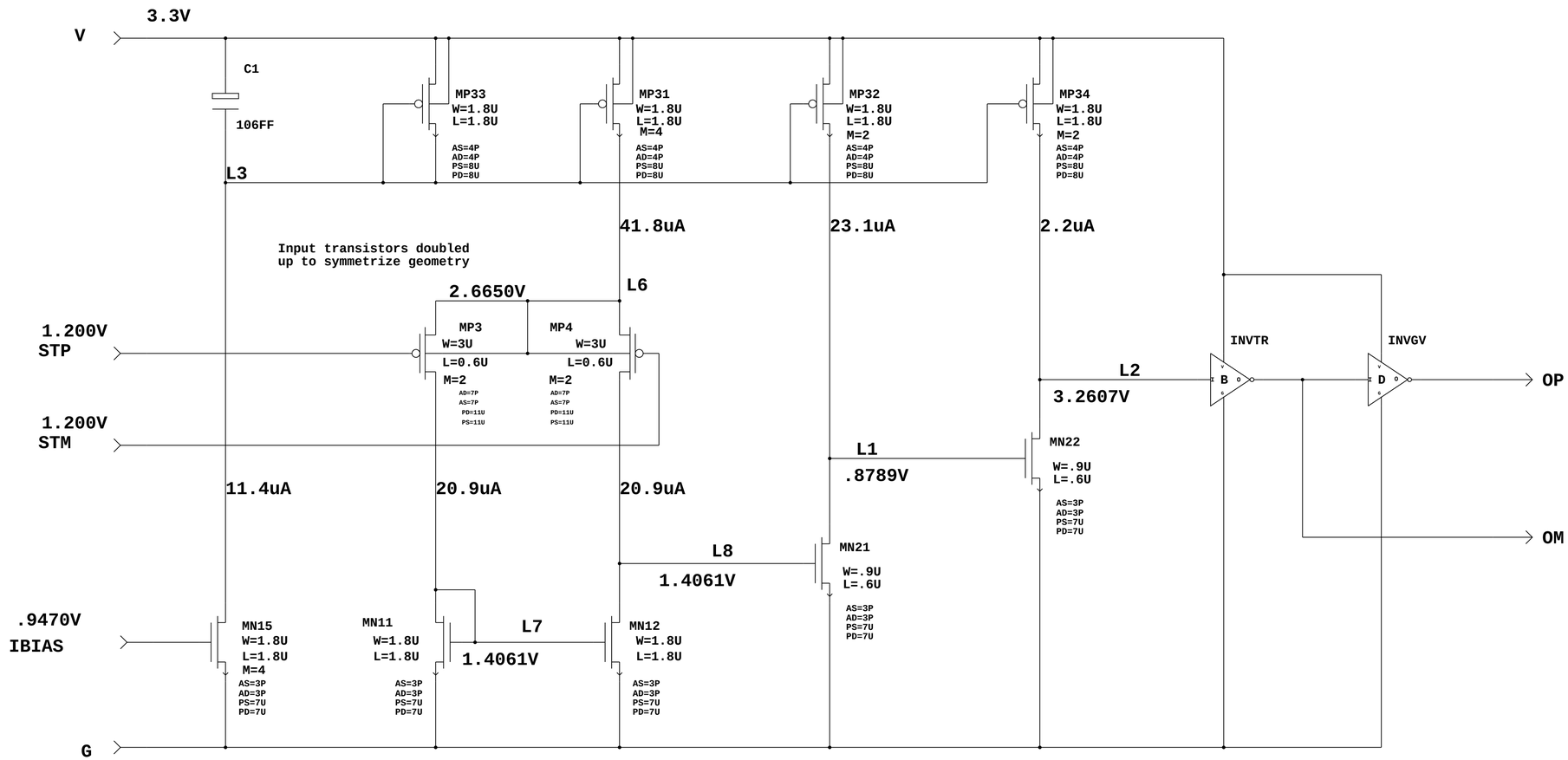
frame9

LAT-DS-03518-50 D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE

Changed 12/13/02 from rcvrf:

Transistors MP3 and MP4 L=.6U from 1.8U



Threshold scan (STM=1.2):

STP	1.1997	1.1996	
L1	1.3058	1.4817	(G= 1760)
L2	2.7781	.4196	(G=23600)
L8	1.3899	1.3845	(G= 54)
OP	3.3	0	

gcfe4: MN15 M=4, MP31 M=4 -> I=14u

gcfe2: MN15 M=3, MP31 M=2 -> I=6u

gcfe3: same as gcfe2

STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	02-26-01	
DFTR	D. Freytag		
CHKR			

SHEET 2 OF

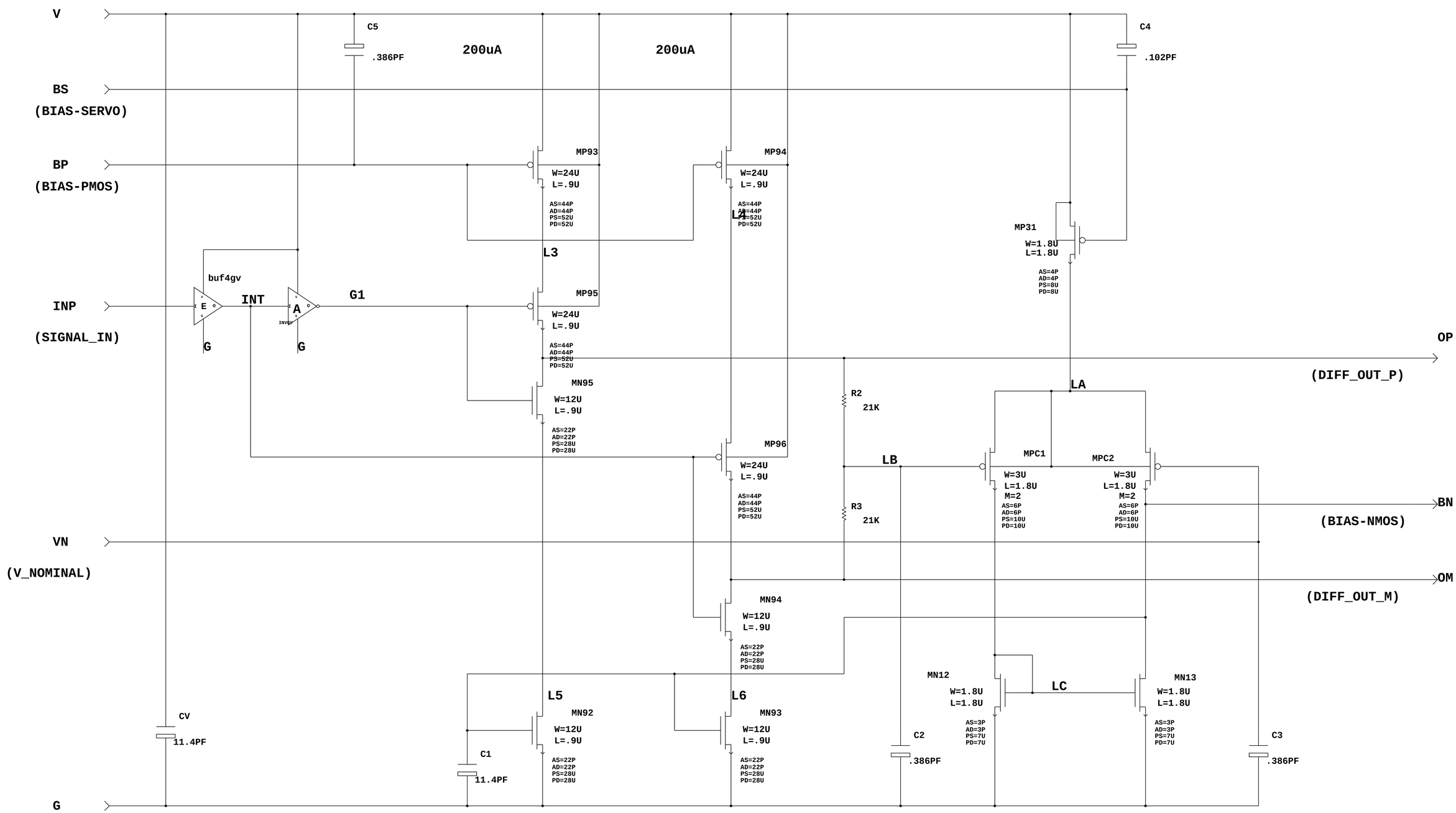
rcvrn

12-13-2002_17:26

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE

Servo Amplifier



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	02-26-01	
DFTR	D. Freytag		
CHKR			

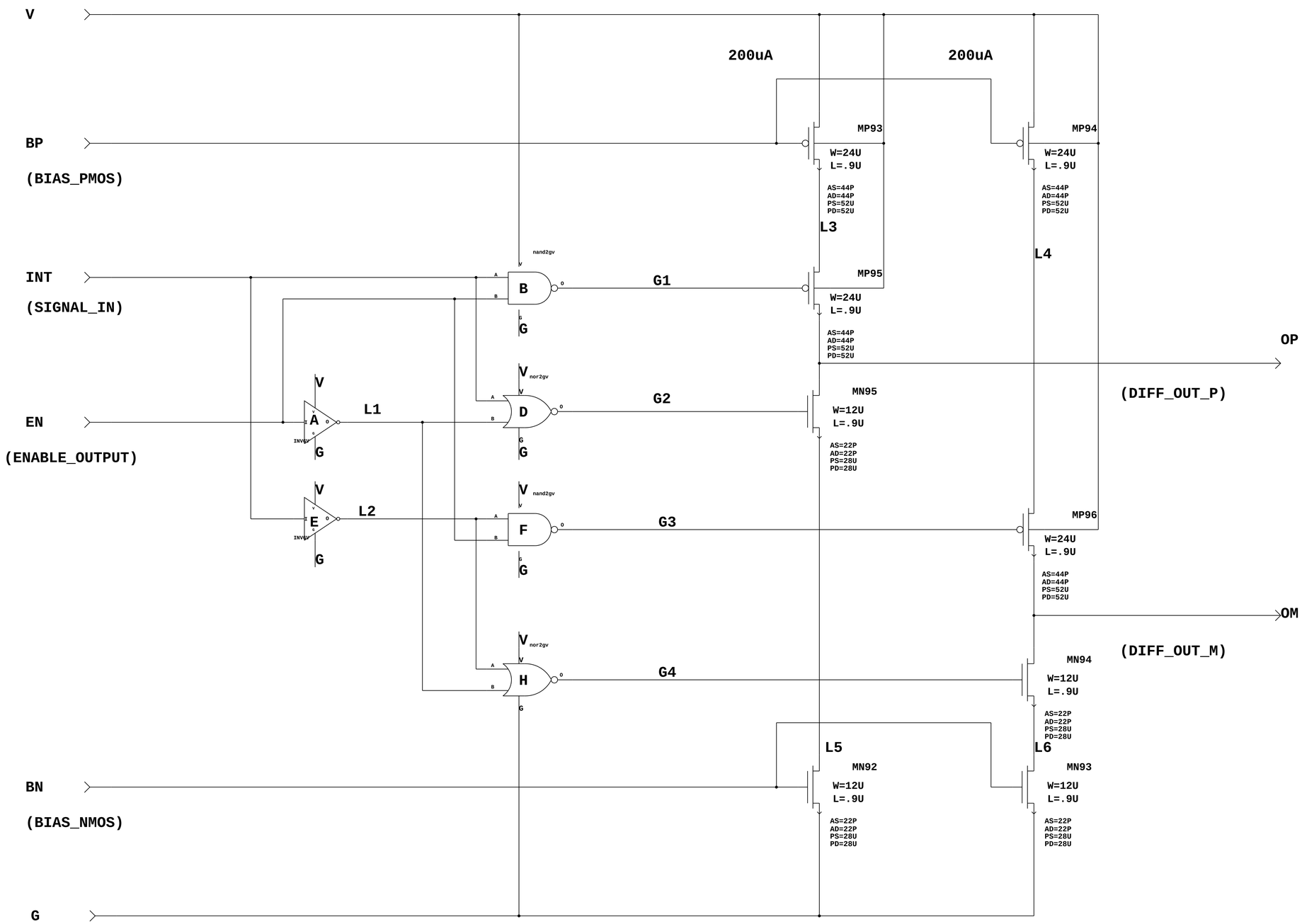
SHEET OF

lvds_drv

4-10-2002_12:54

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	02-26-01	
DFTR	D. Freytag		
CHKR			

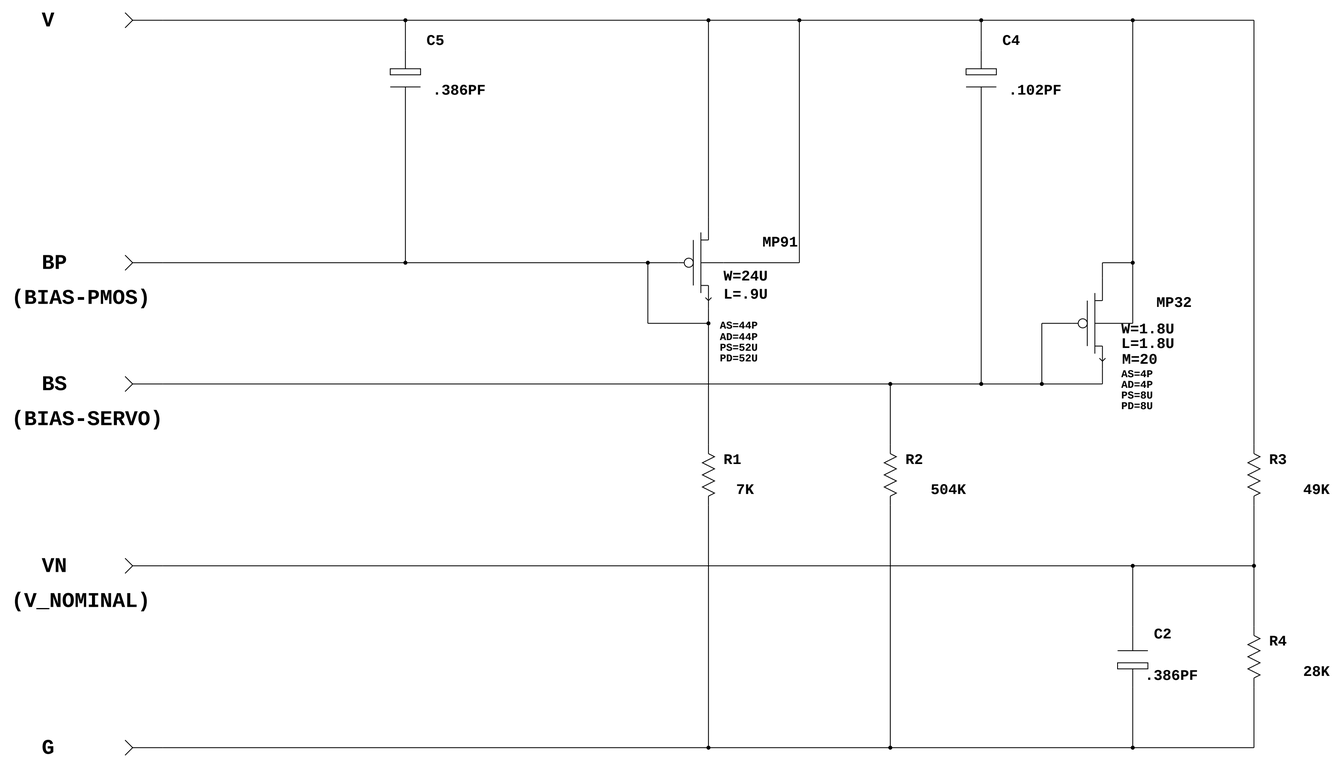
SHEET OF

lvds_exp

4-10-2002_13:00

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	02-26-01	
DFTR	D. Freytag		
CHKR			

SHEET OF

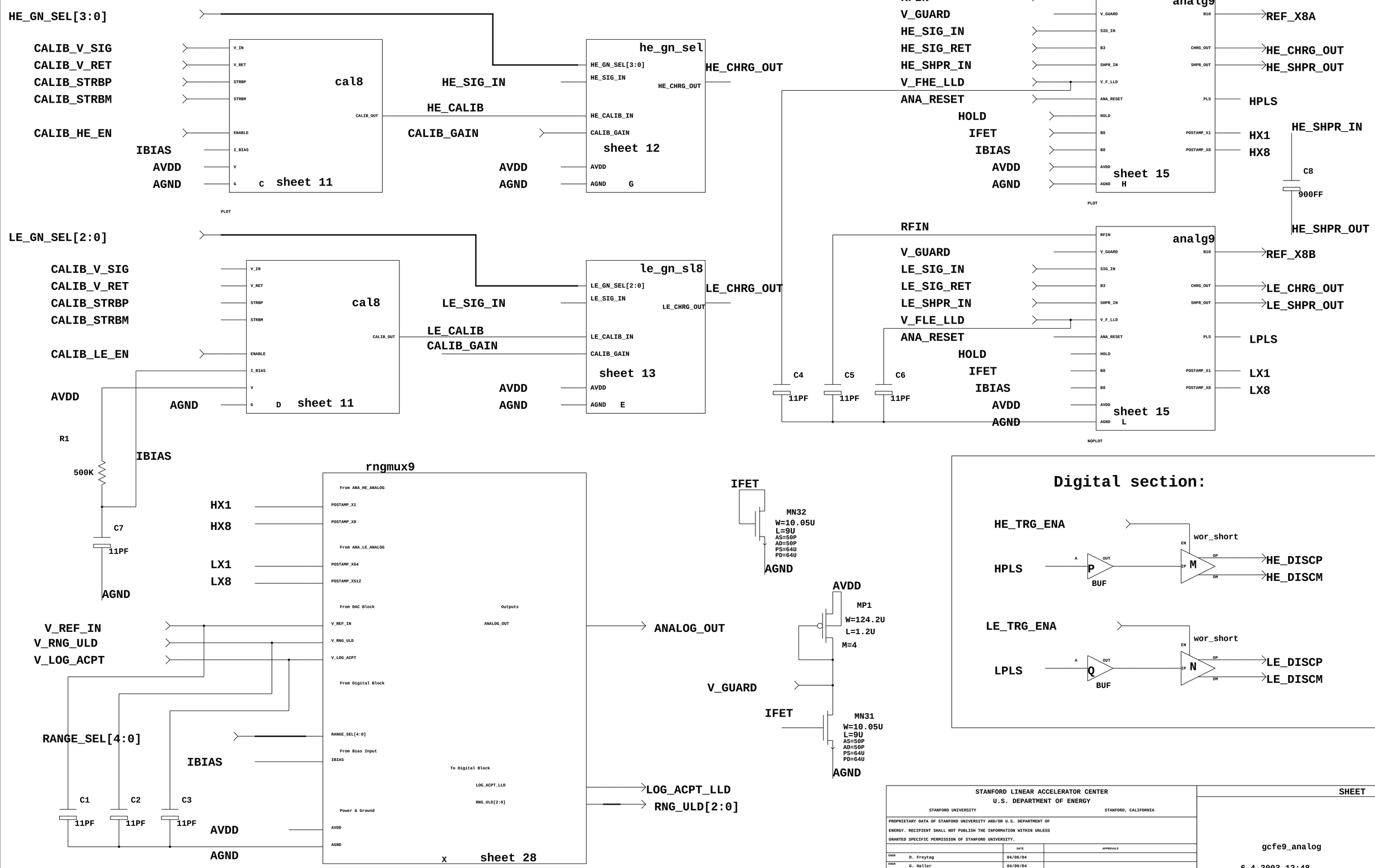
ll_bias

2-12-2002_16:13

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE

gcfe9a: added C8=900FF in HE shaper



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

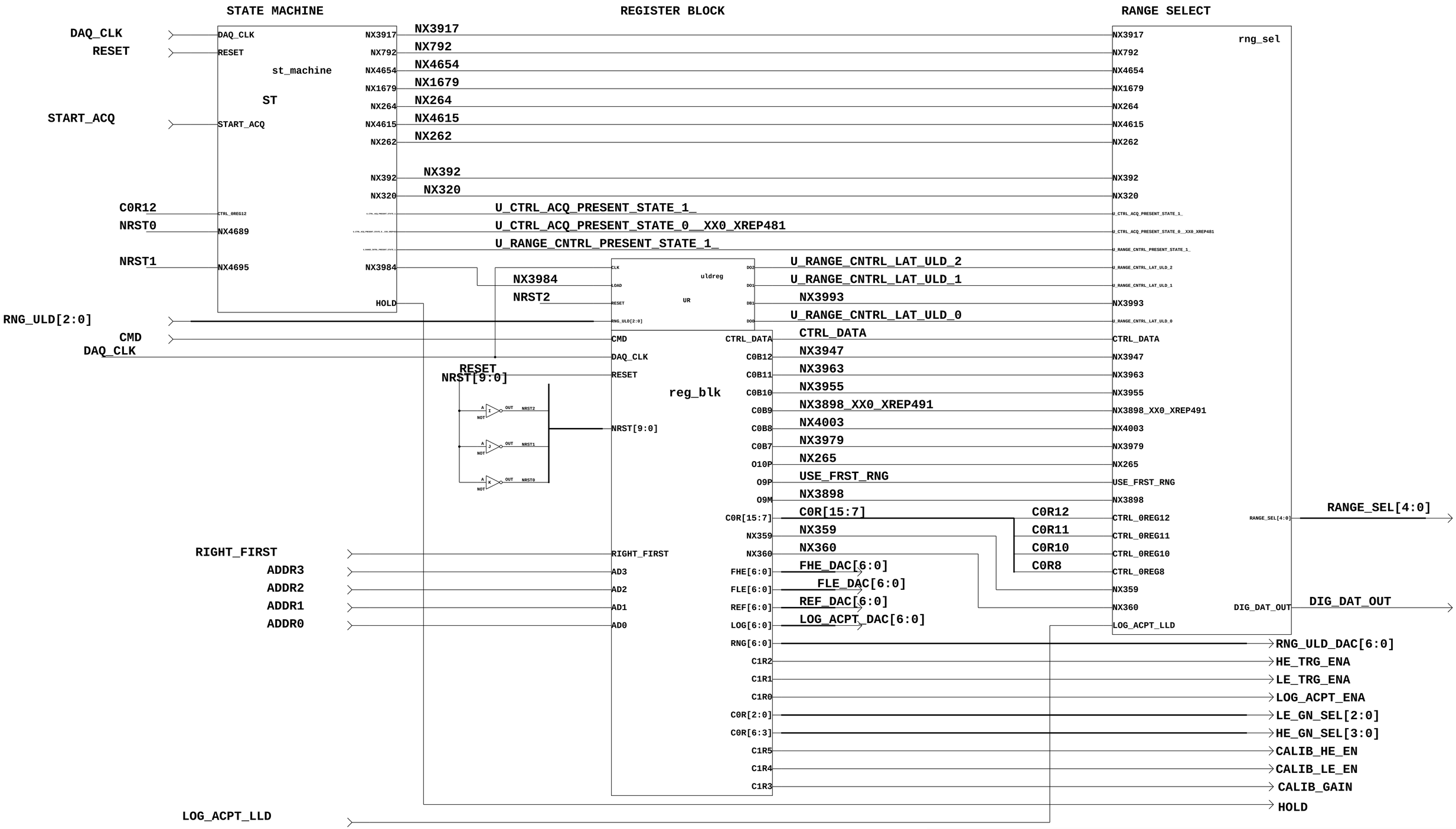
SHEET 10 OF

gcfe9_analog

6-4-2003_12:48

LAT-DS-03518-50 D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

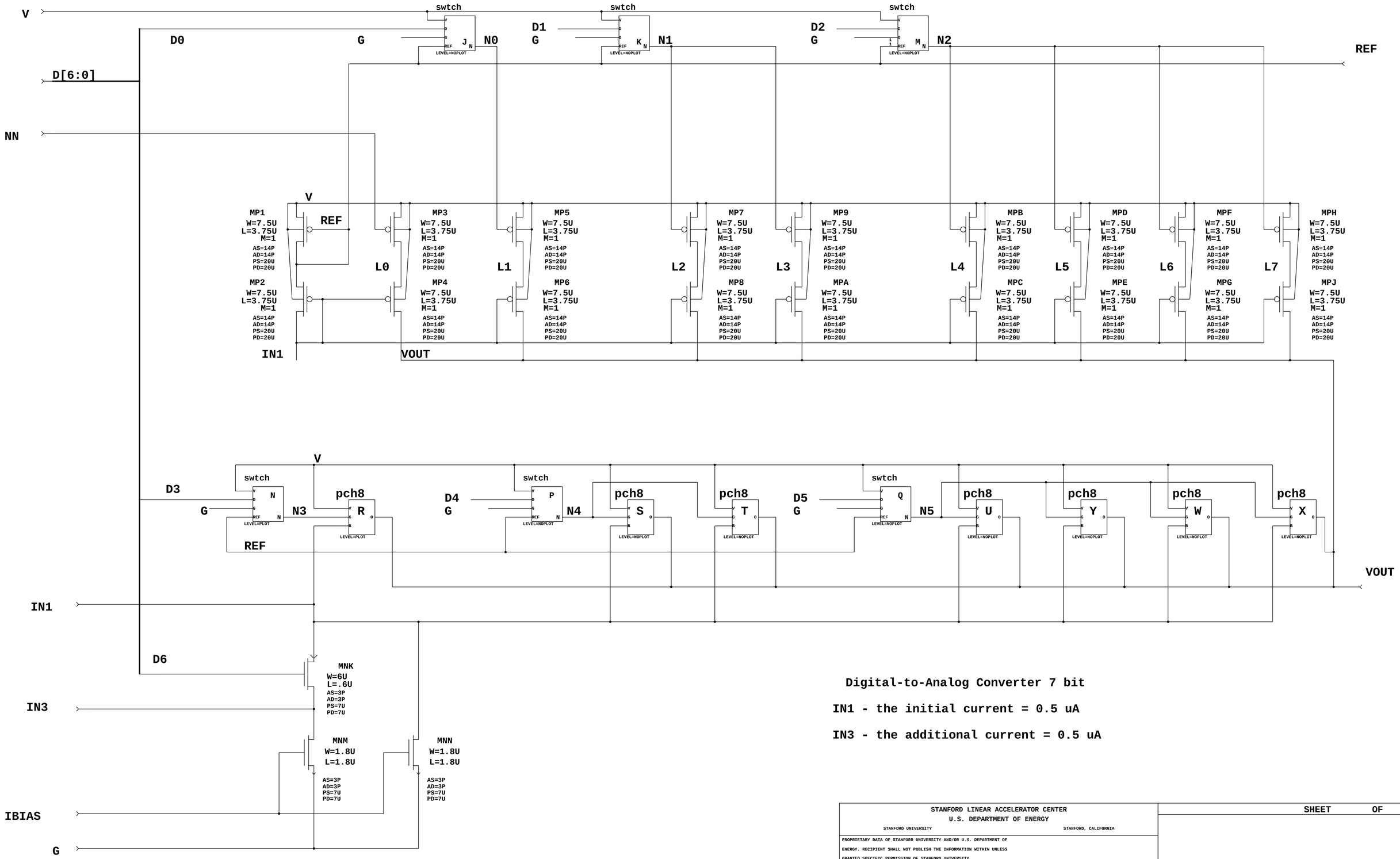
SHEET OF

digital

4-5-2004_16:45

LAT-DS-03518-50 D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



Digital-to-Analog Converter 7 bit

IN1 - the initial current = 0.5 uA

IN3 - the additional current = 0.5 uA

STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET OF

dac

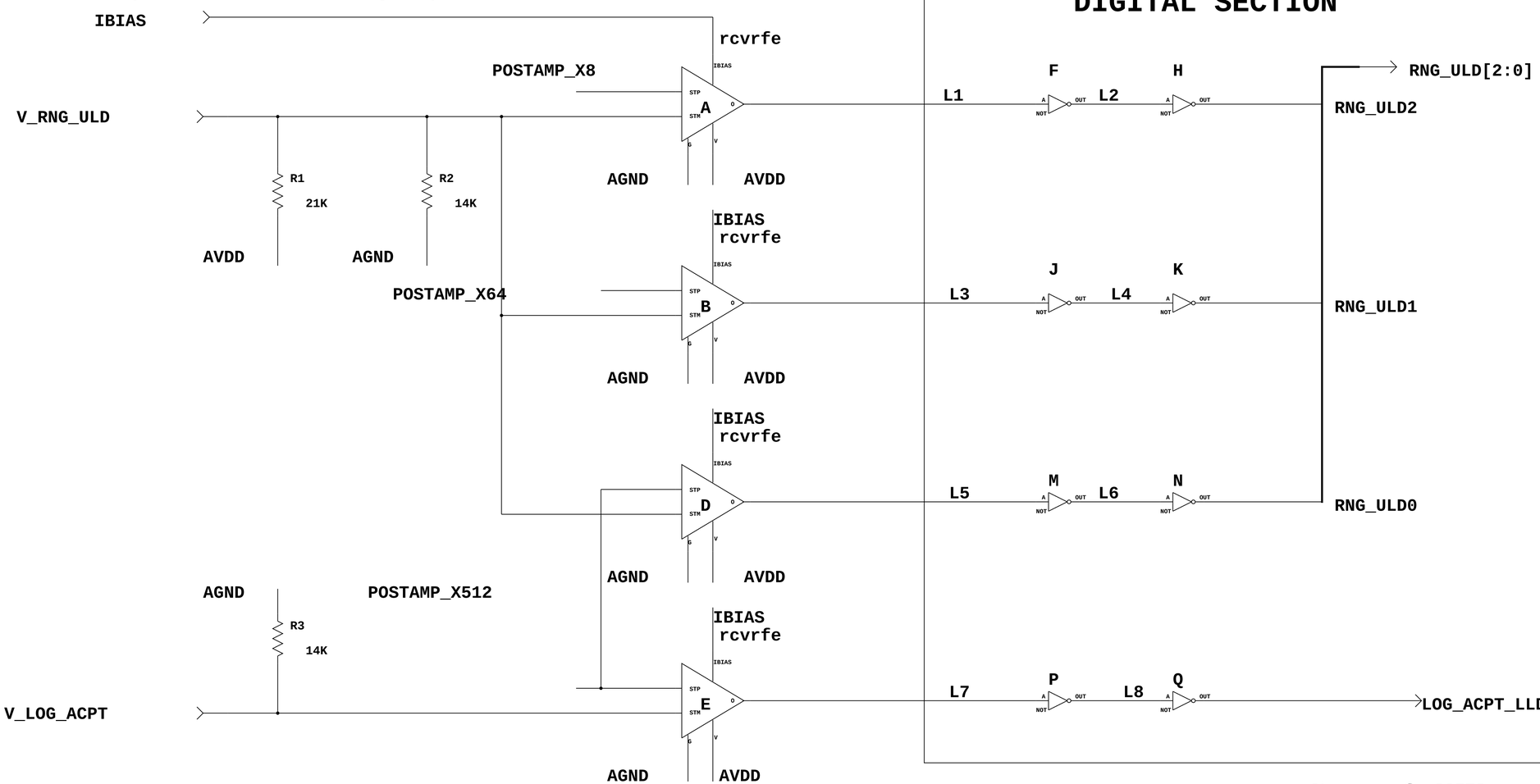
4-16-2003_17:02

LAT-DS-03518-50

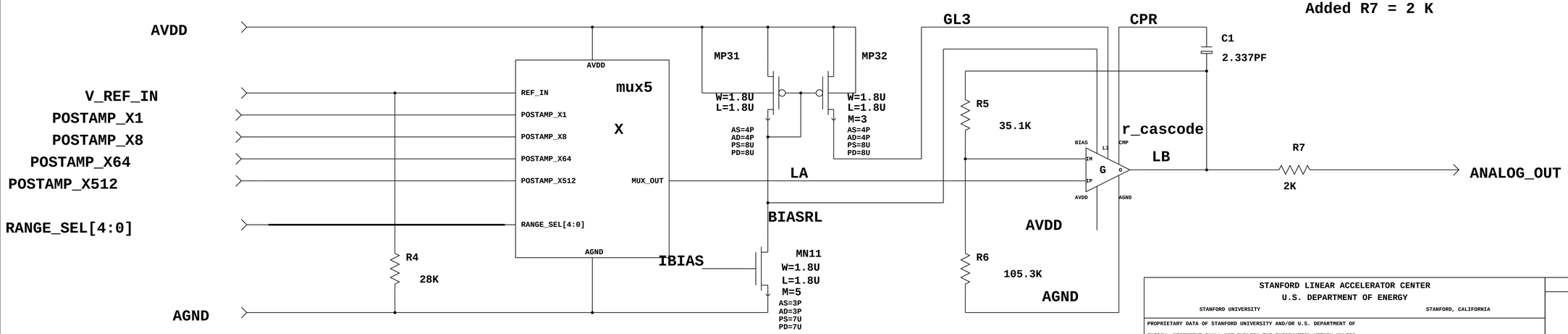
D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE

c.f. gcfe8: Switched values of R1,R2
(Back to values used in gcfe6)



Change c.f. GCFE8:
C1=663FF -> C1=2.337PF
Added R7 = 2 K



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET 28 OF

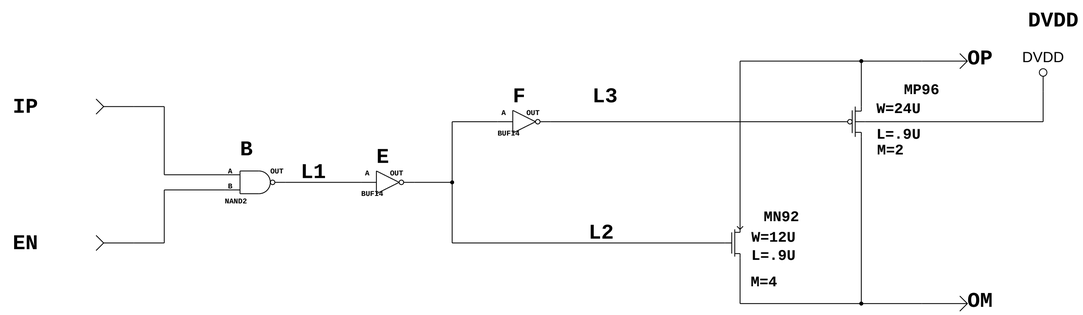
rngmux9

1-15-2003_12:24

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

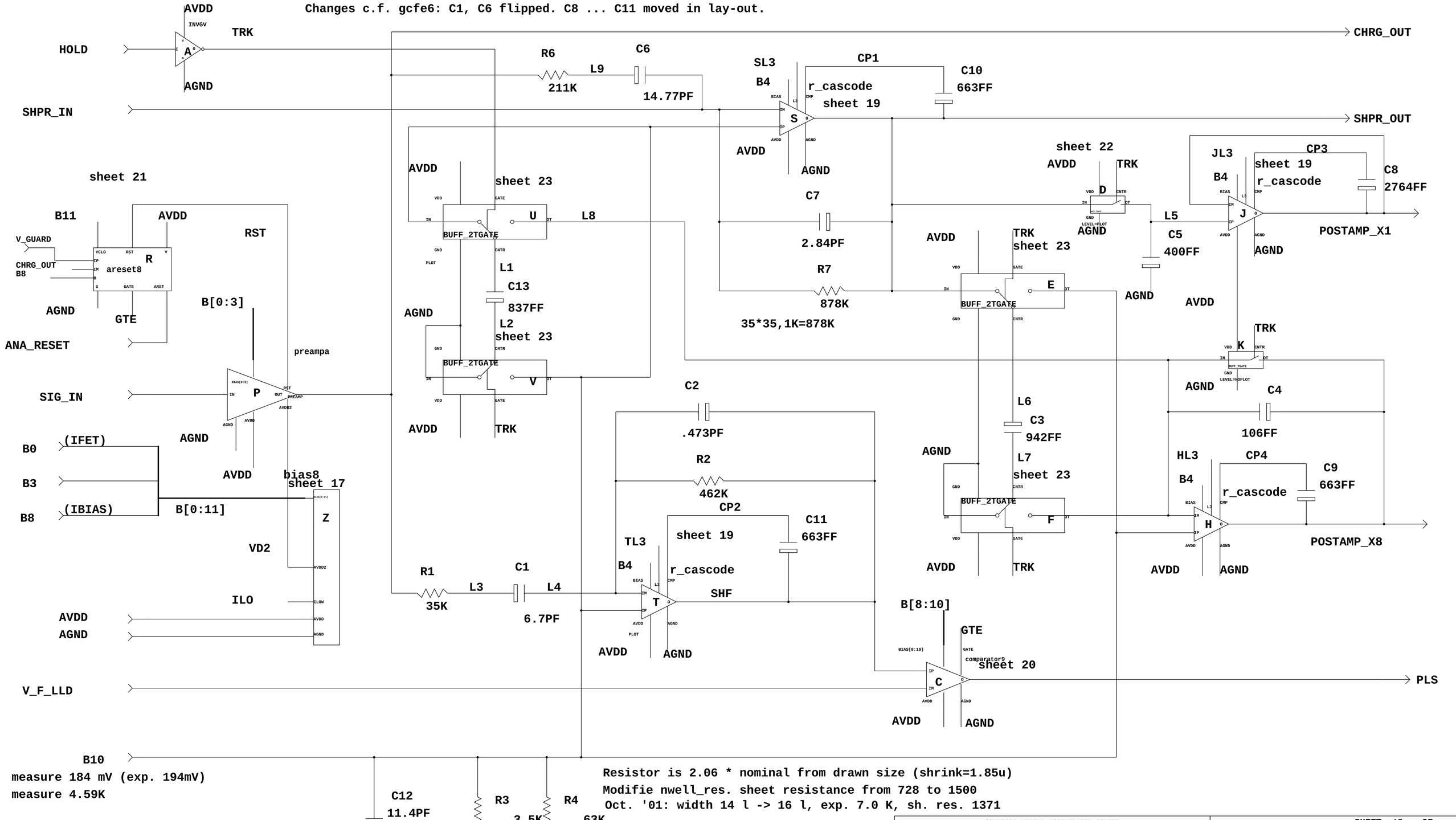
		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

wor_short

LAT-DS-03518-50 D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE

Changes c.f. gcfe6: C1, C6 flipped. C8 ... C11 moved in lay-out.



Resistor is 2.06 * nominal from drawn size (shrink=1.85u)
Modifie nwell_res. sheet resistance from 728 to 1500
Oct. '01: width 14 l -> 16 l, exp. 7.0 K, sh. res. 1371

STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY — STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

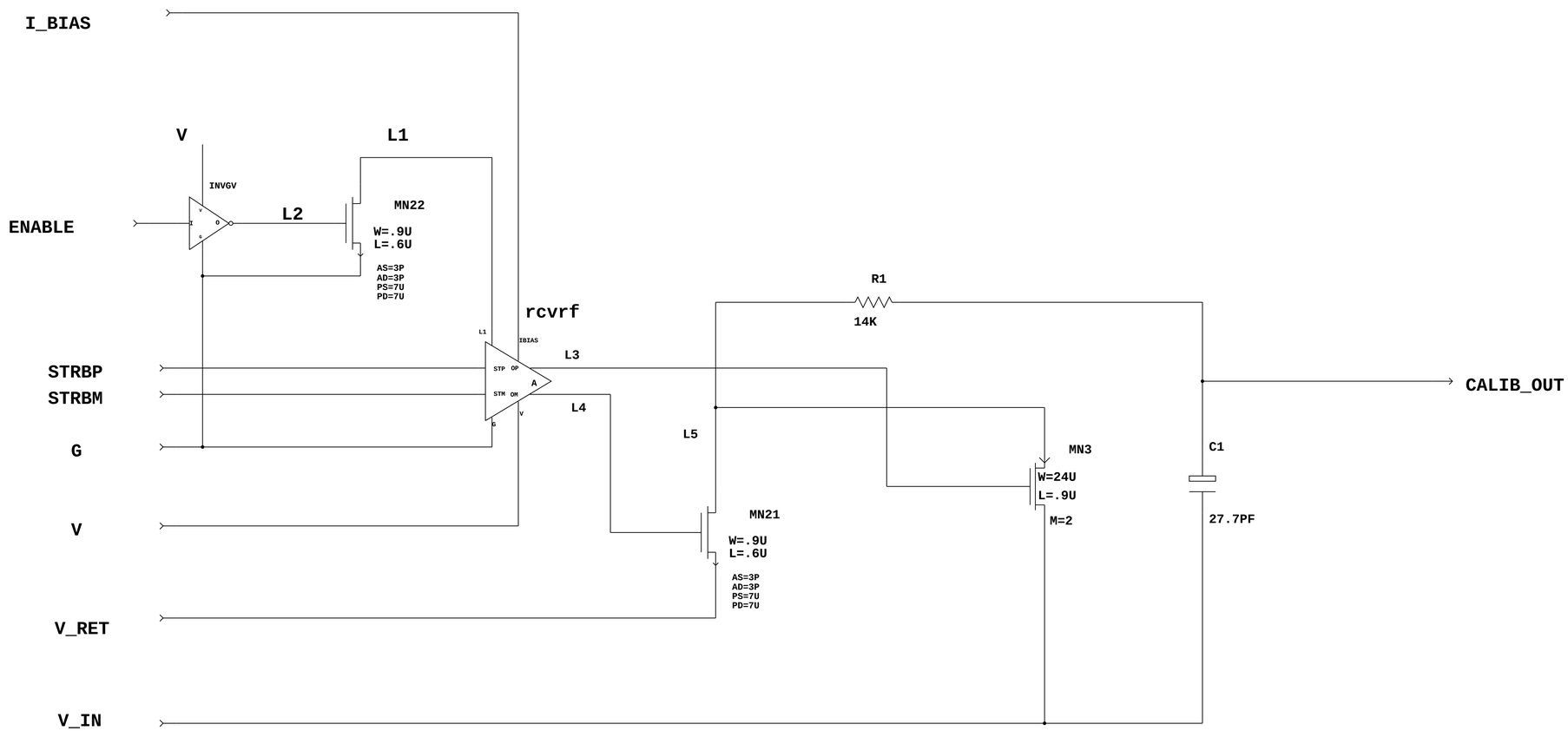
analg9

6-2-2003_14:57

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET 11 OF

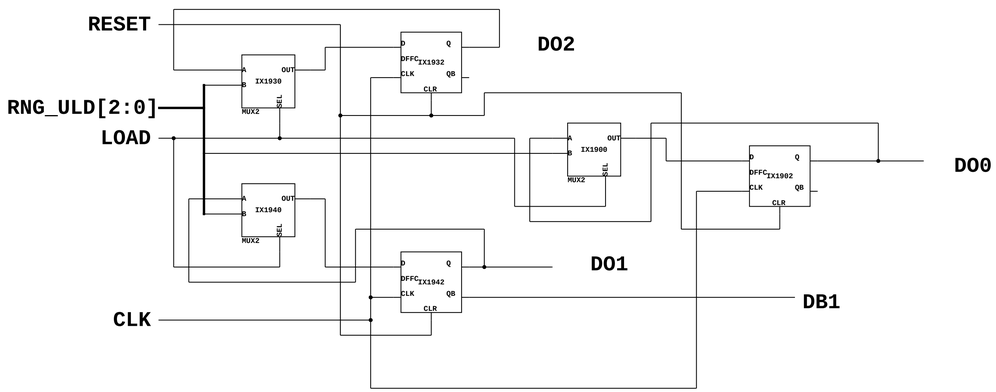
cal8

4-17-2003_12:54

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET OF

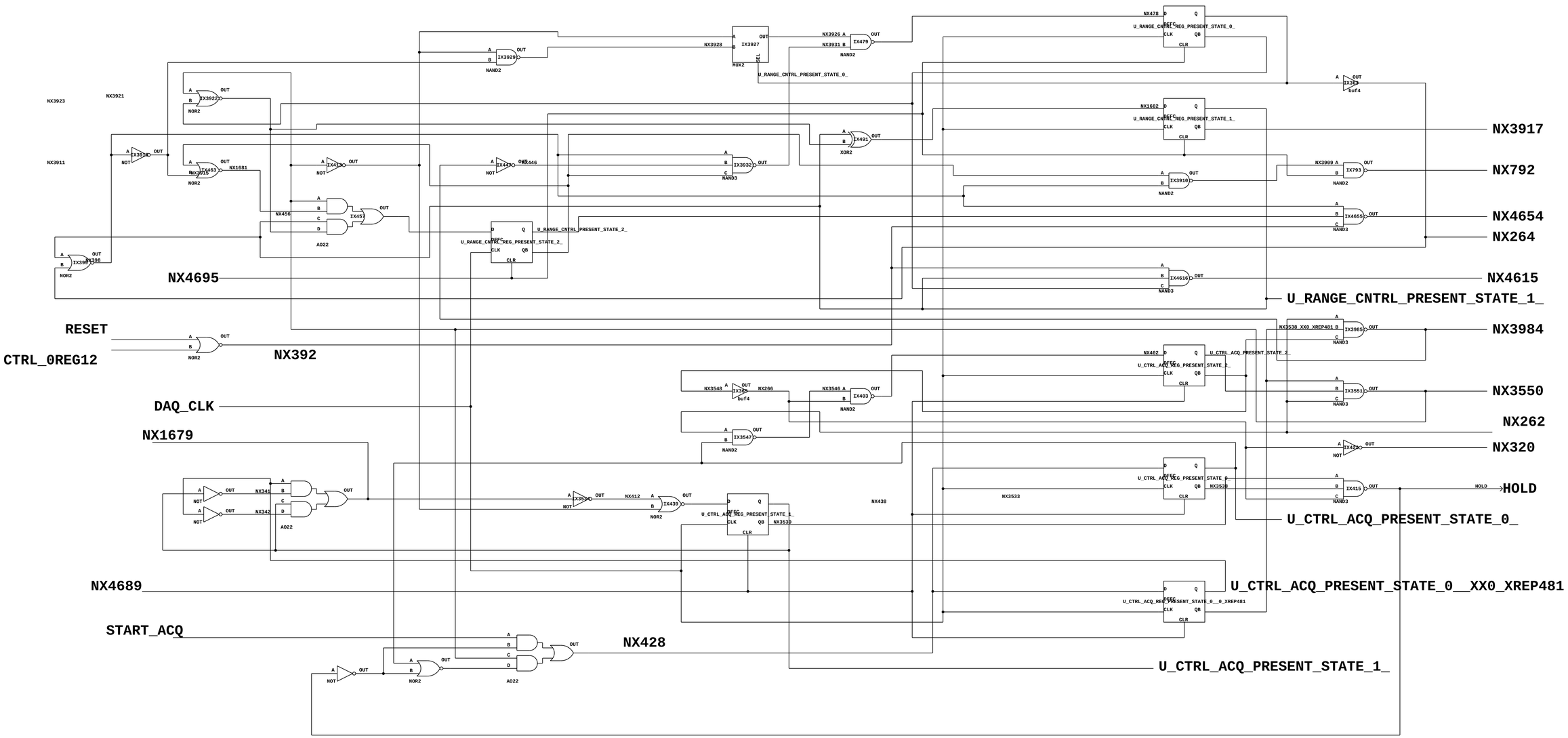
uldreg

4-6-2004_12:13

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY — STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET OF

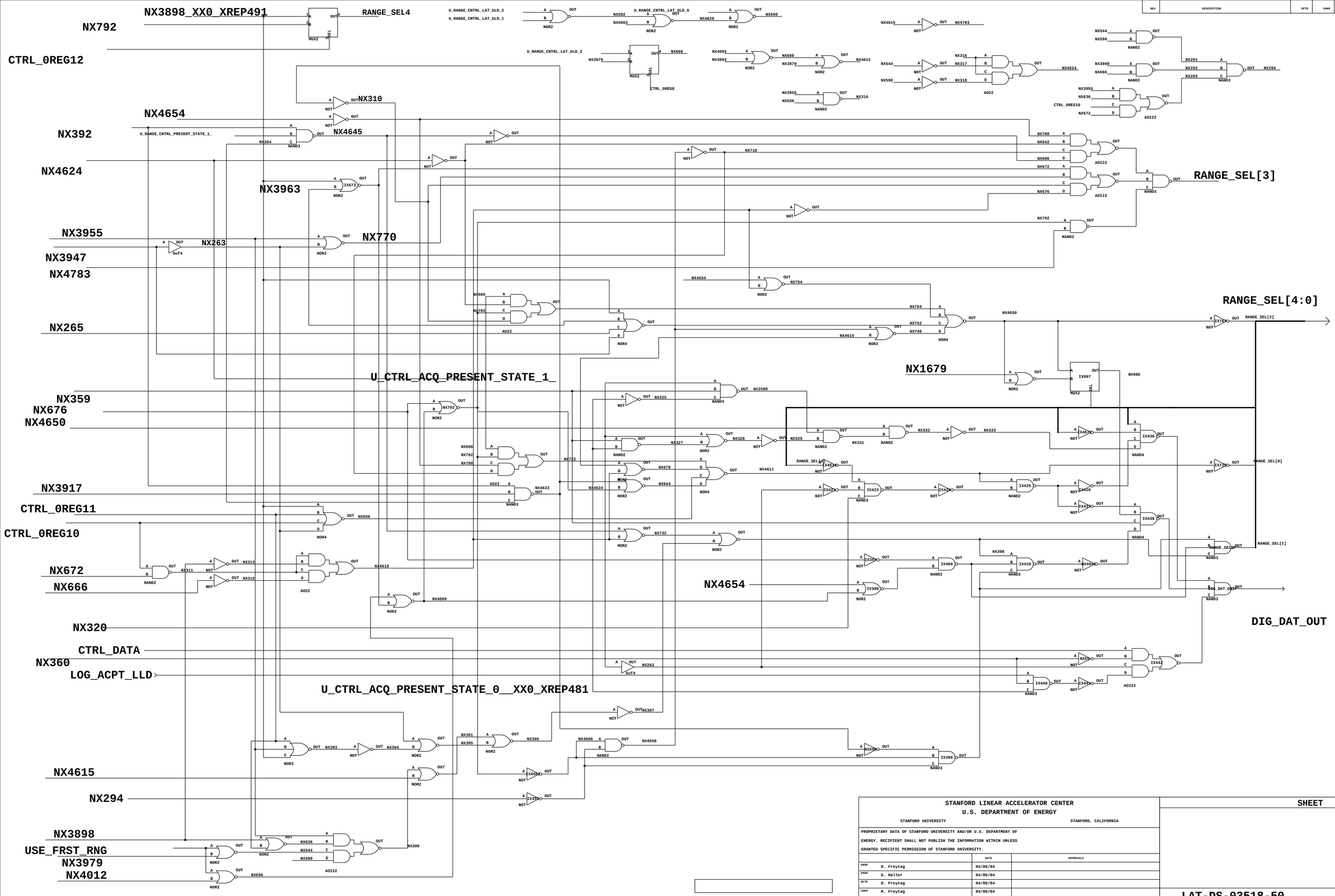
st_machine

4-6-2004_11:35

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

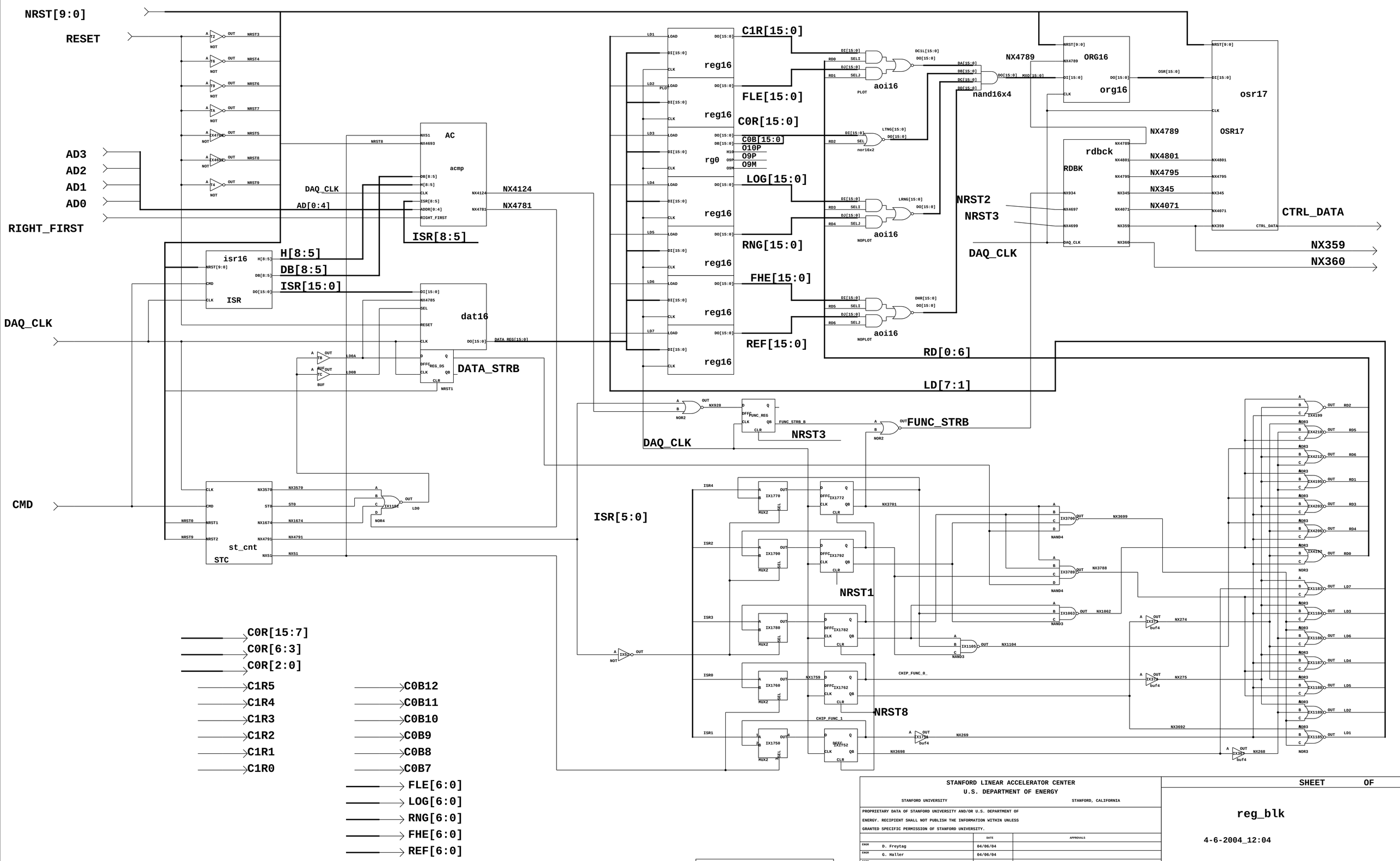
PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET OF

LAT-DS-03518-50 D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY

STANFORD UNIVERSITY — STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET OF

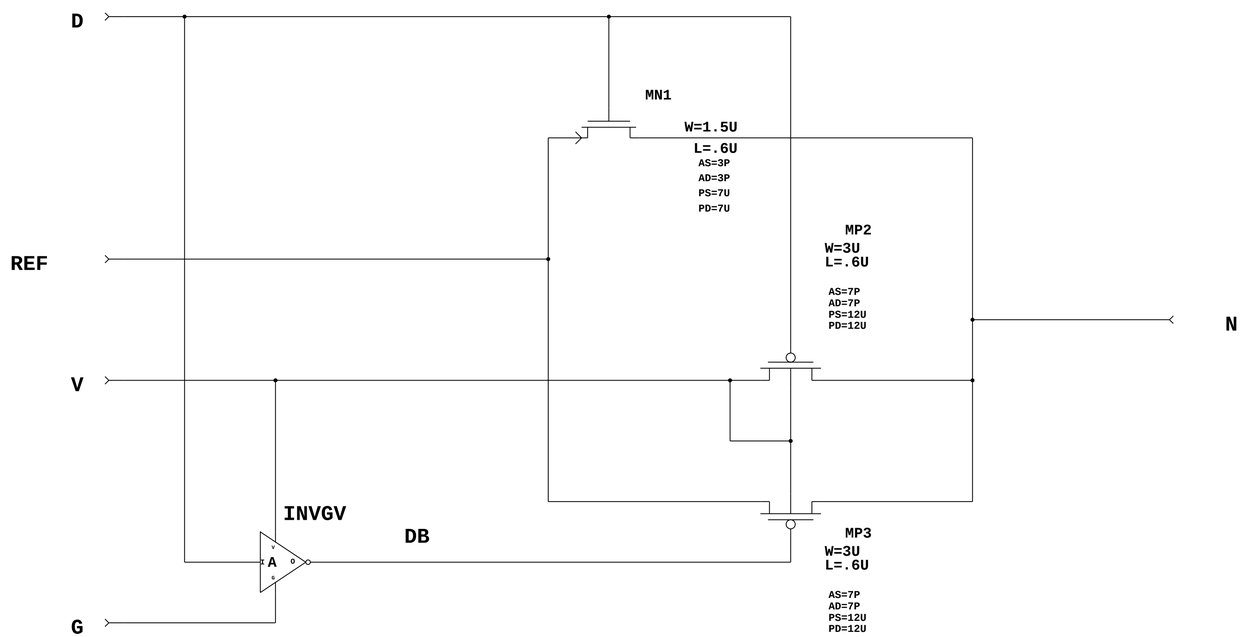
reg_blk

4-6-2004_12:04

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET OF

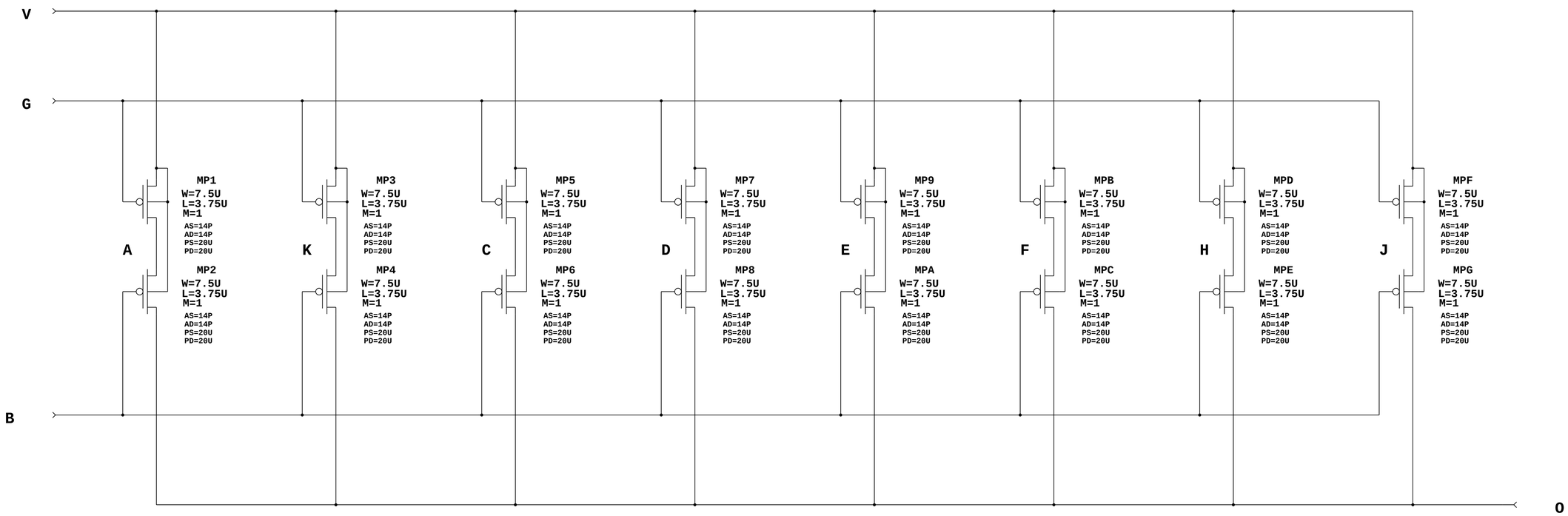
swtch

4-16-2003_16:58

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY

STANFORD UNIVERSITY — STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET OF

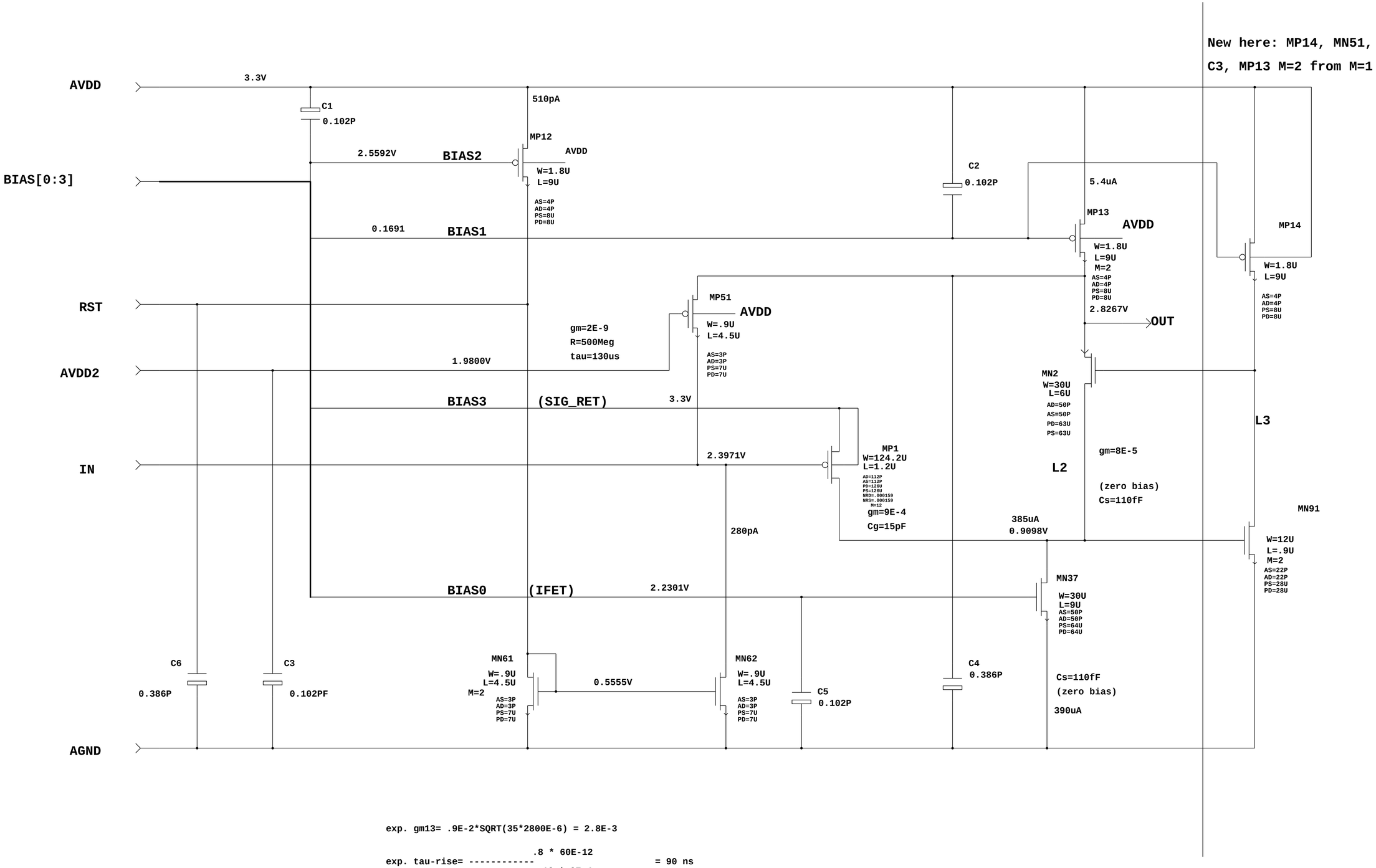
pch8

4-16-2003_16:50

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



exp. gm13= .9E-2*SQRT(35*2800E-6) = 2.8E-3

$$\text{exp. tau-rise} = \frac{.8 * 60E-12}{.13 * 9E-4} = 90 \text{ ns}$$

STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

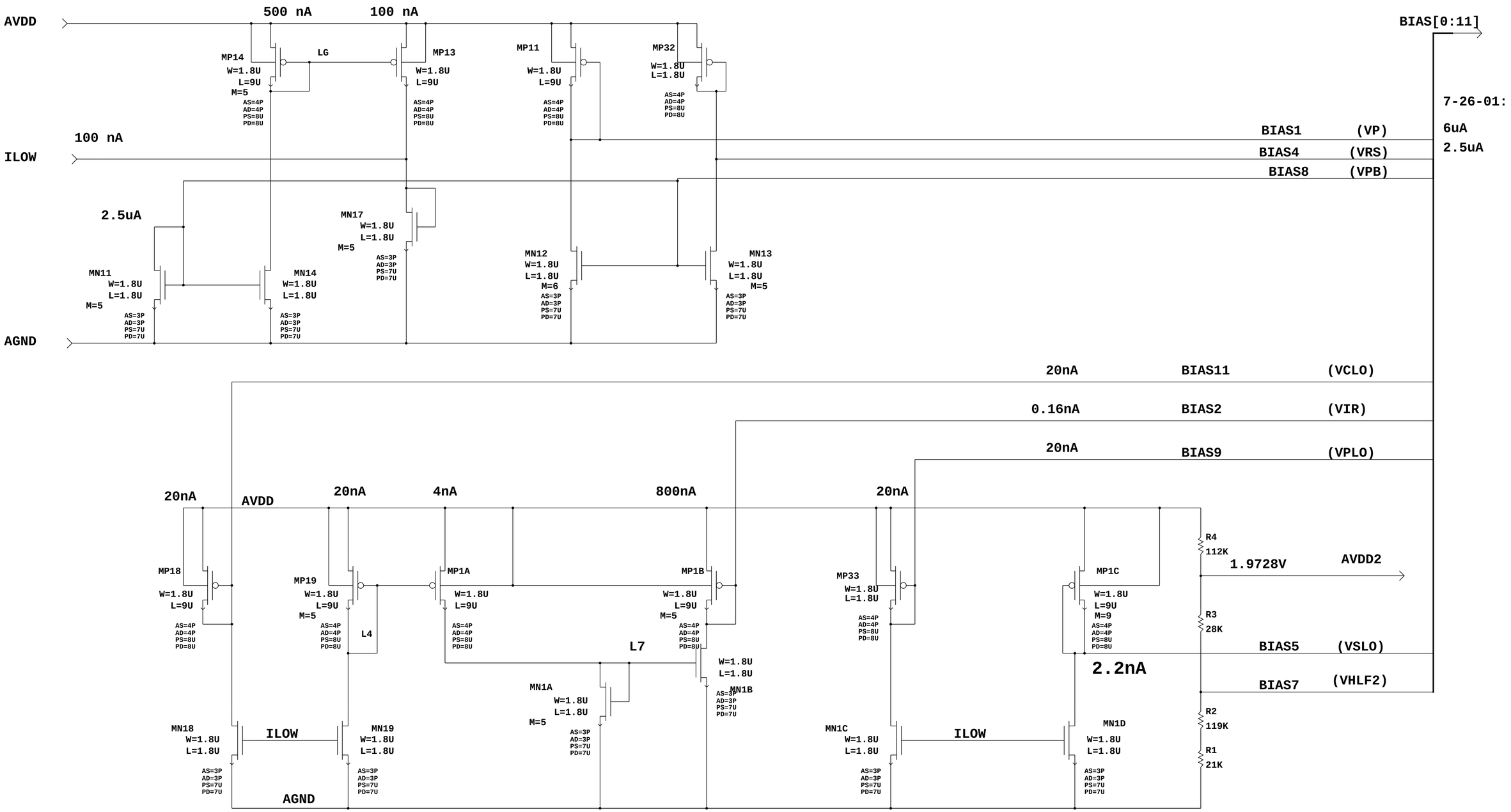
preampa

1-28-2002_14:52

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



bias0 = IFET

bias3 = QVDD

bias10= VREF

STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET 17 OF

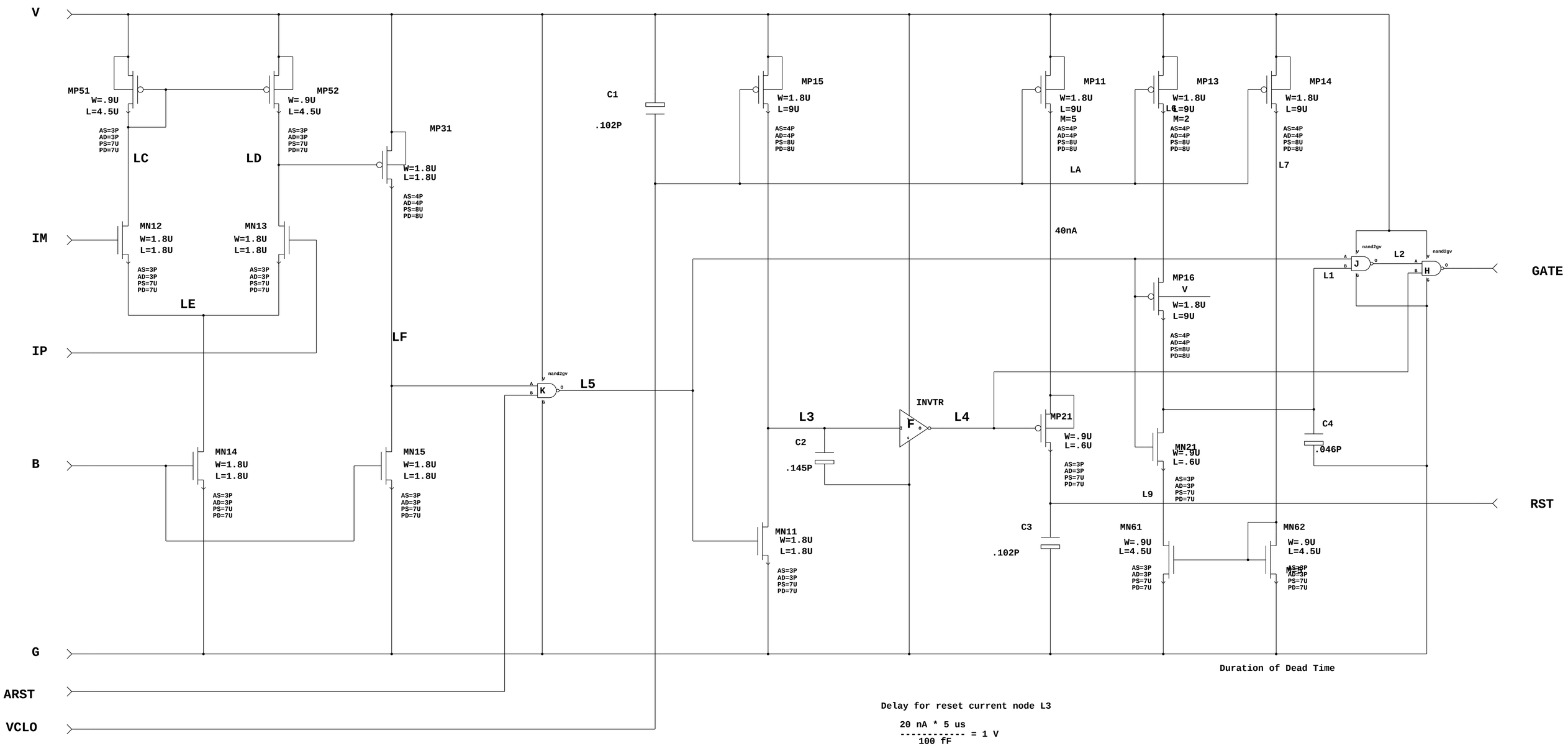
bias8

12-17-2002_12:18

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



Delay for reset current node L3

$$\frac{20\ \text{nA} * 5\ \text{us}}{100\ \text{fF}} = 1\ \text{V}$$

STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

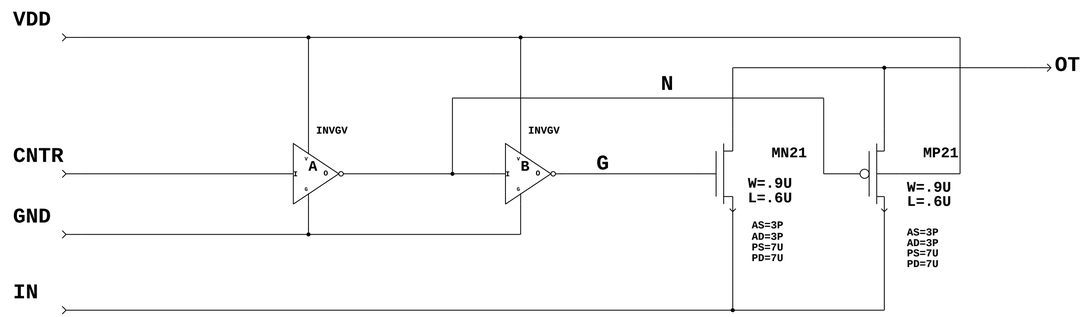
areset8

5-21-2003_14:30

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

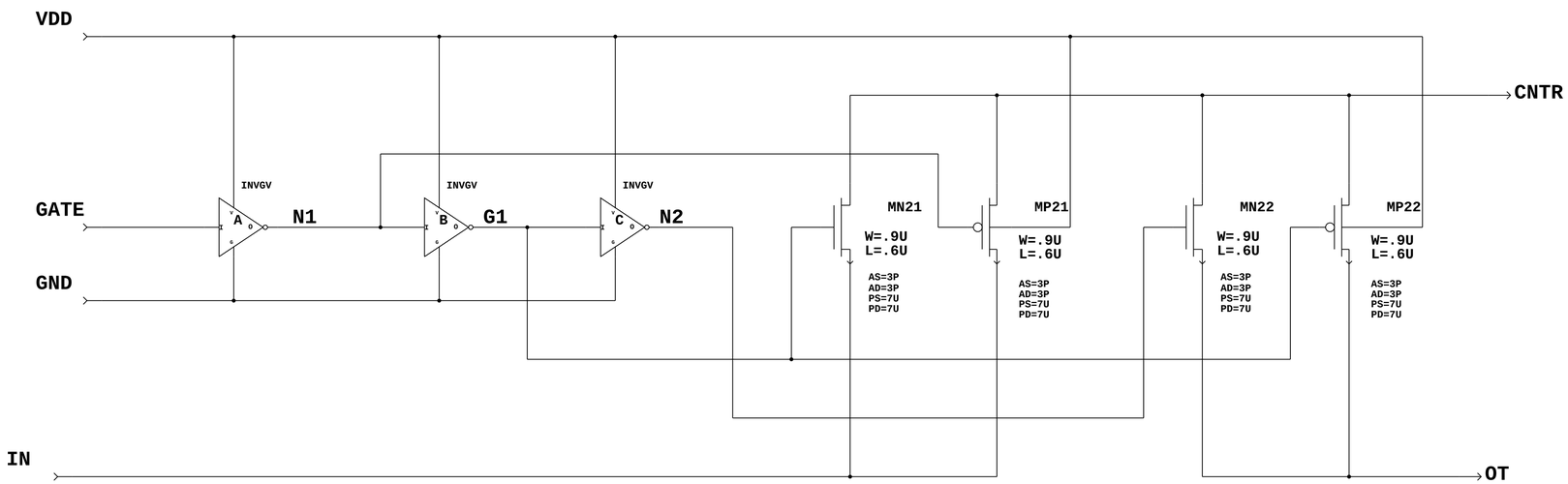
PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

buff_tgate
5-21-2001_13:40

LAT-DS-03518-50 D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

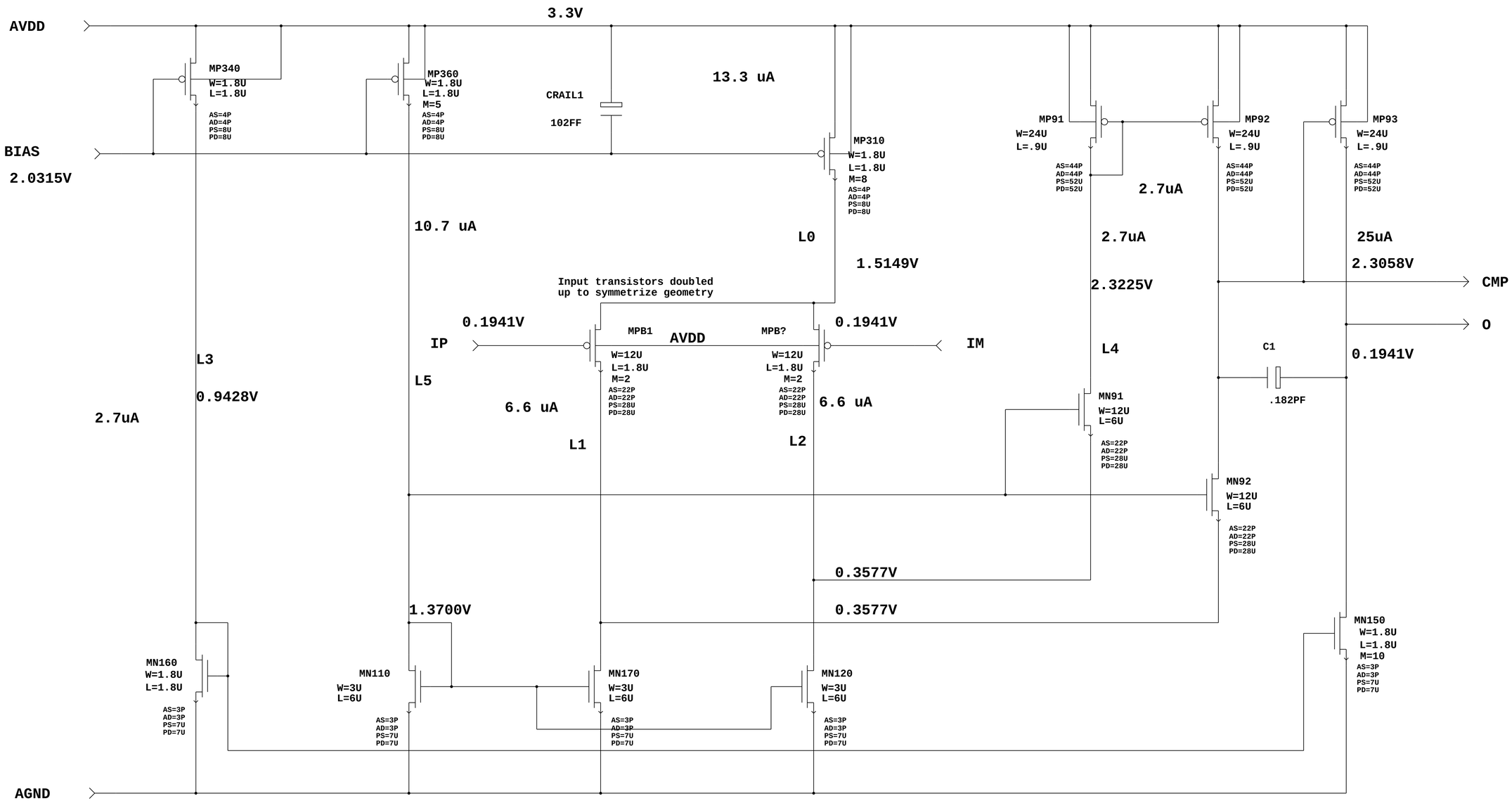
		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

buff_2tgate
11-21-2001_17:47

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

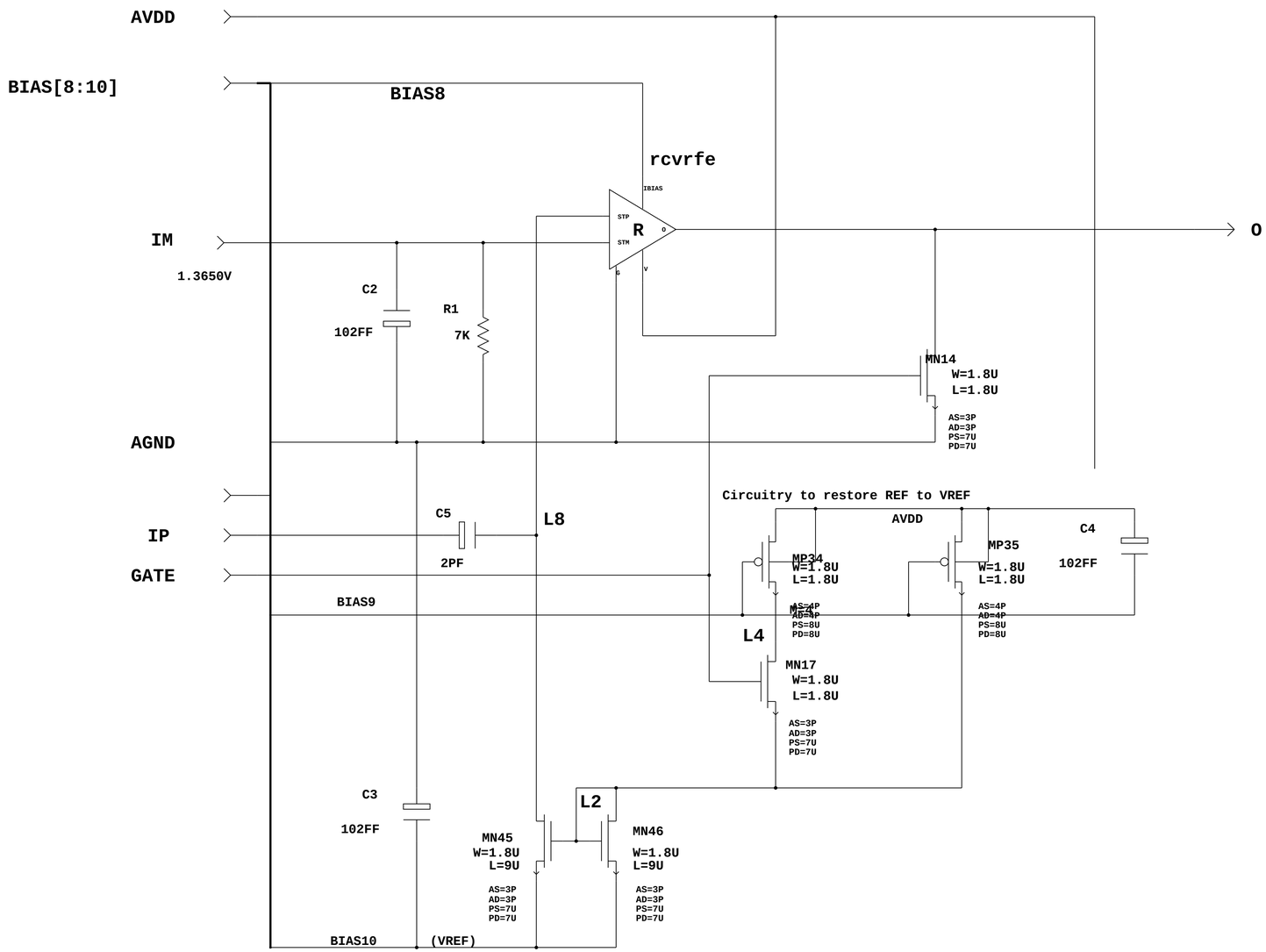
r_cascode

11-7-2001_11:28

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET 20 OF

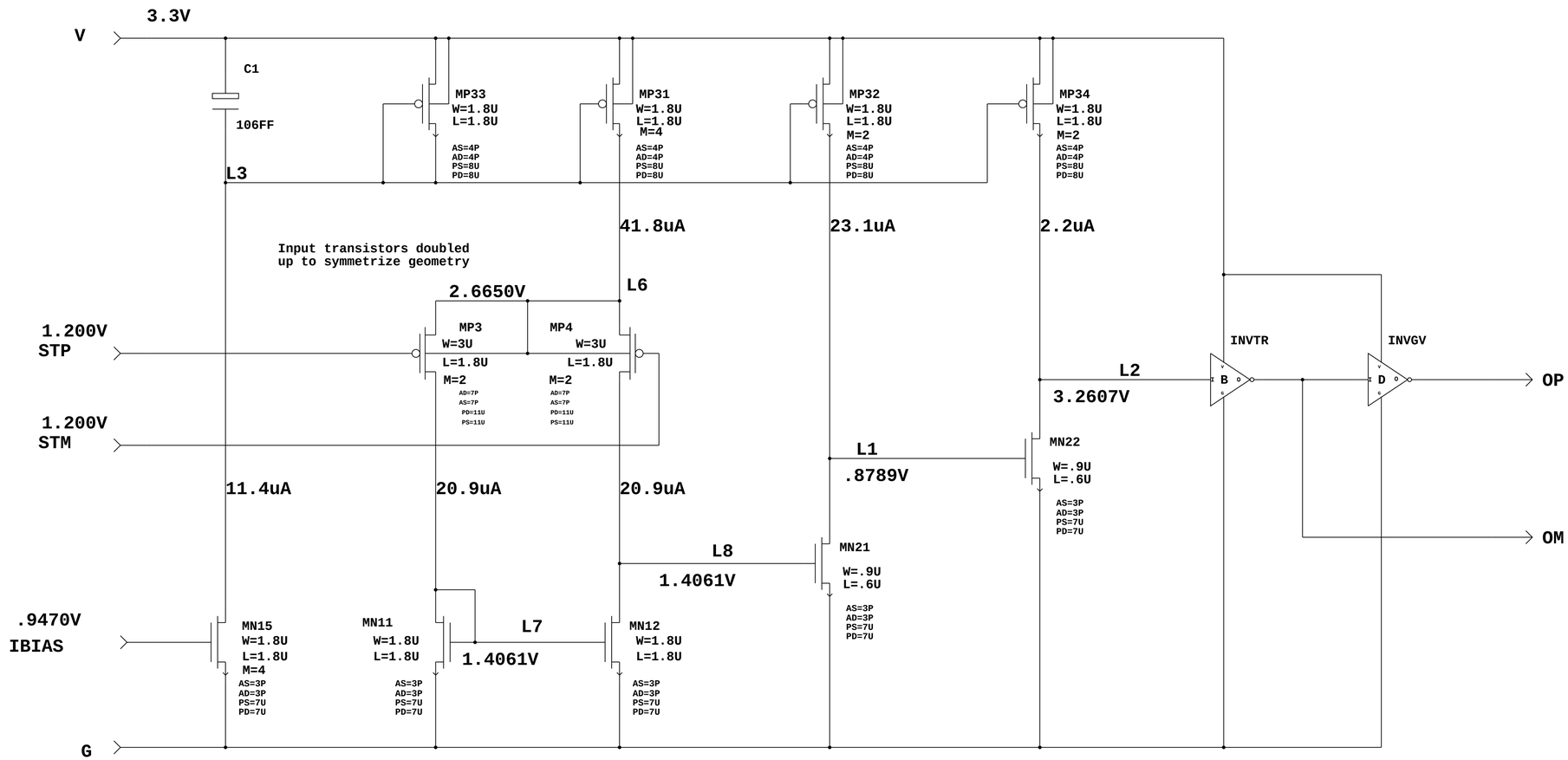
comparator9

6-2-2003_14:49

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



Threshold scan (STM=1.2):

STP	1.1997	1.1996	
L1	1.3058	1.4817	(G= 1760)
L2	2.7781	.4196	(G=23600)
L8	1.3899	1.3845	(G= 54)
OP	3.3	0	

gcfe4: MN15 M=4, MP31 M=4 -> I=14u

gcfe2: MN15 M=3, MP31 M=2 -> I=6u

gcfe3: same as gcfe2

STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

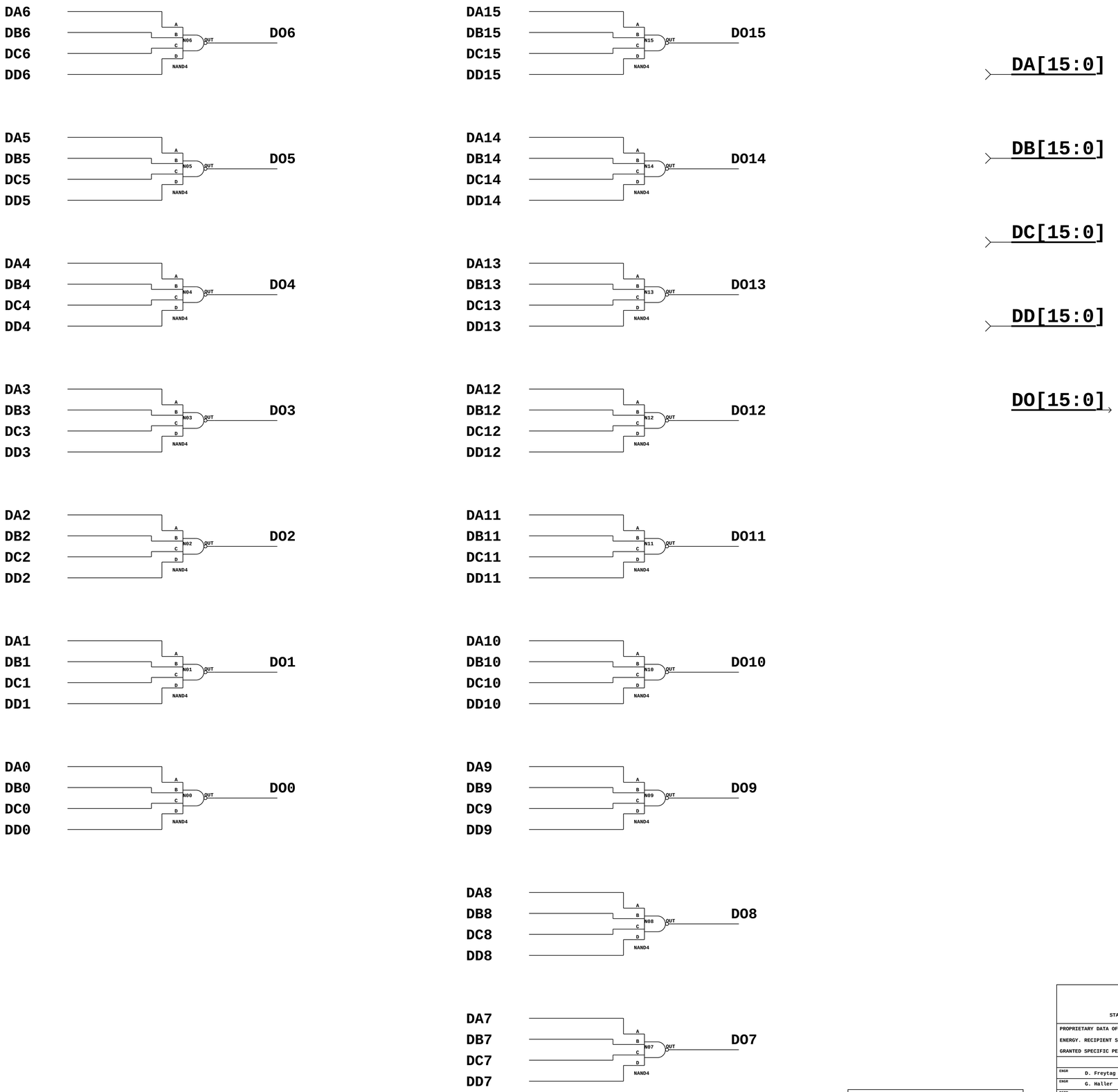
PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	02-26-01	
DFTR	D. Freytag		
CHKR			

rcvrf
4-26-2002_14:32

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET OF

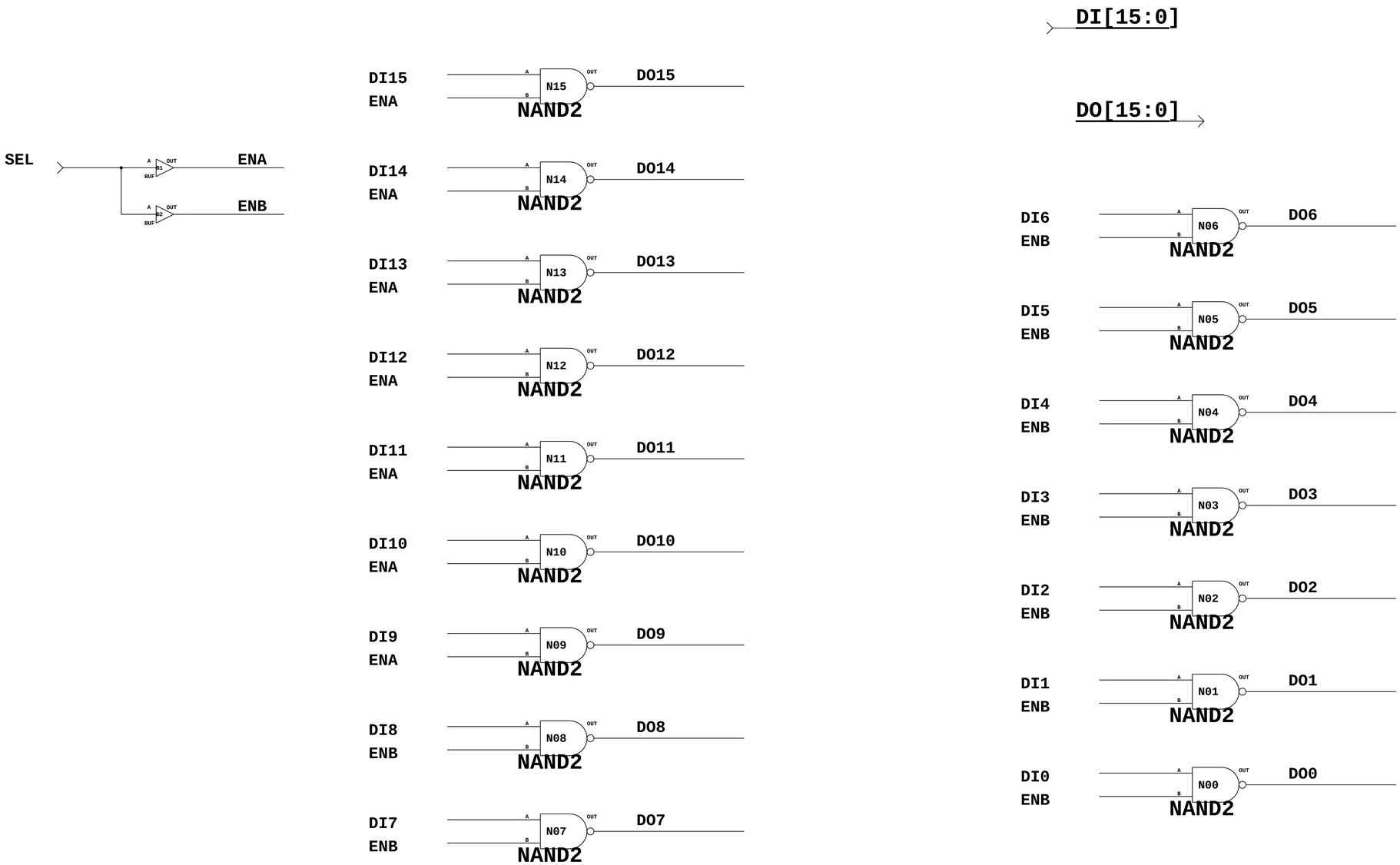
nand16x4

4-10-2003_14:17

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET OF

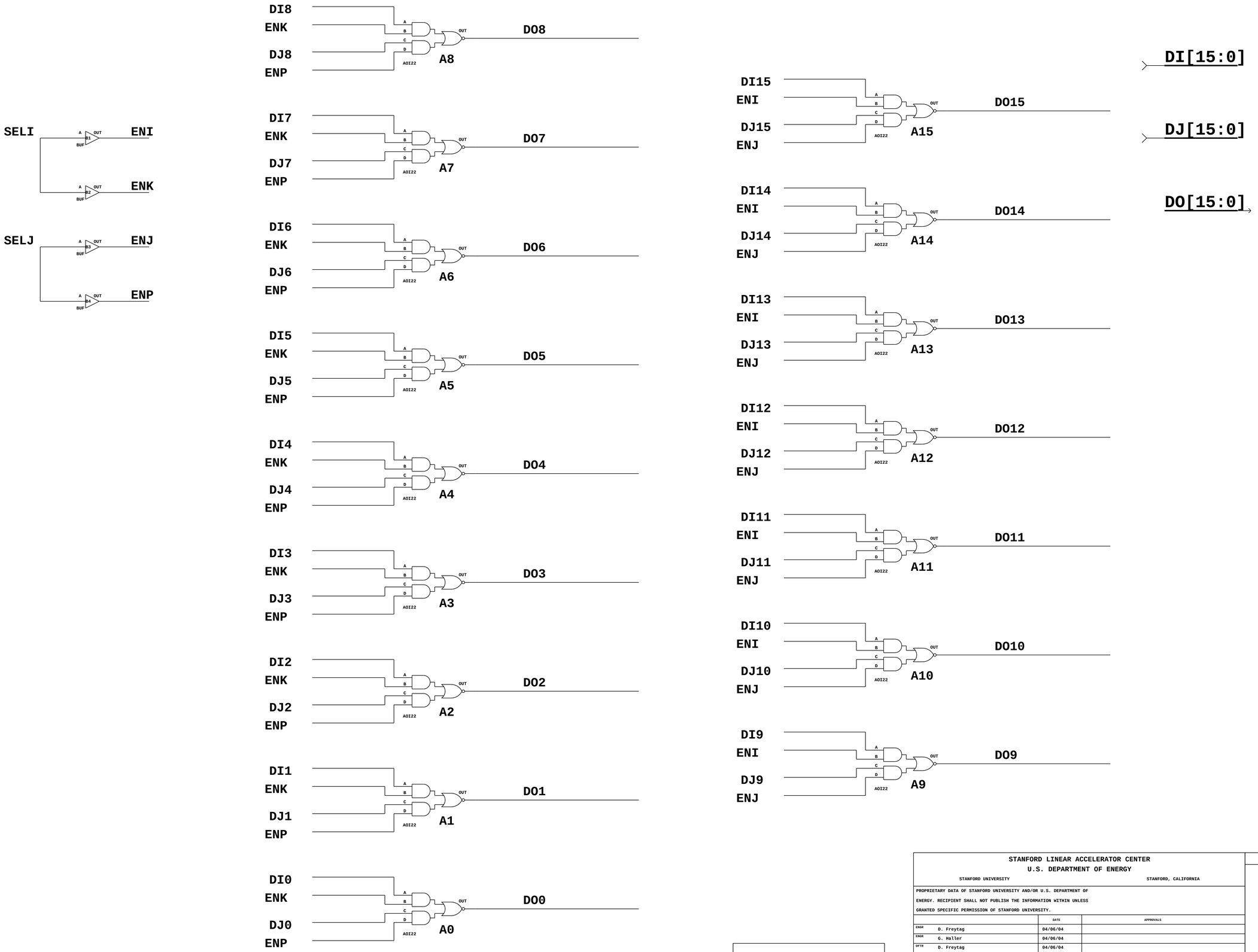
nor16x2

4-10-2003_14:15

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET OF

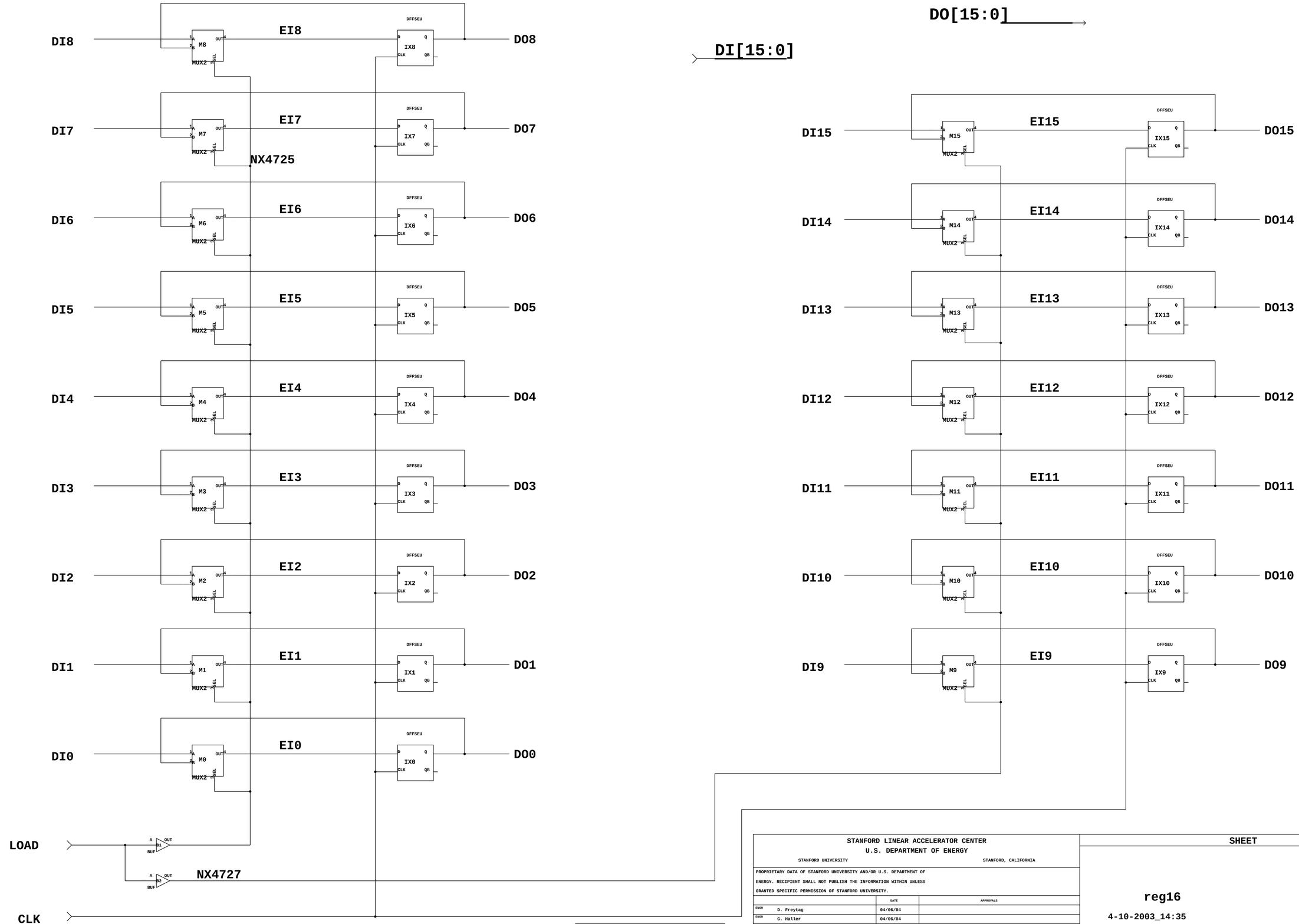
aoi16

4-10-2003_14:14

LAT-DS-03518-50

D

REV	DESCRIPTION	DFTR	CHKR	ENGR APPV	DATE



STANFORD LINEAR ACCELERATOR CENTER
U.S. DEPARTMENT OF ENERGY
STANFORD UNIVERSITY STANFORD, CALIFORNIA

PROPRIETARY DATA OF STANFORD UNIVERSITY AND/OR U.S. DEPARTMENT OF ENERGY. RECIPIENT SHALL NOT PUBLISH THE INFORMATION WITHIN UNLESS GRANTED SPECIFIC PERMISSION OF STANFORD UNIVERSITY.

		DATE	APPROVALS
ENGR	D. Freytag	04/06/04	
ENGR	G. Haller	04/06/04	
DFTR	D. Freytag	04/06/04	
CHKR	M. Freytag	04/06/04	

SHEET OF

reg16

4-10-2003_14:35

LAT-DS-03518-50

D

GLAST LAT SPECIFICATION	Document # **LAT-DS-03518-50**	Date Effective 12 April 2004
	Prepared by(s) Jim Ampe	Supersedes None
	Subsystem/Office Calorimeter Subsystem	
Document Title **Calorimeter Front End ASIC – VHDL Code**		

Gamma-ray Large Area Space Telescope (GLAST)

Large Area Telescope (LAT)

Calorimeter GCFE9A Flight ASIC VHDL Code

DOCUMENT APPROVAL

Prepared by:

James Ampe Naval Research Lab	Date

Approved by:

Gunther Haller LAT Electronics Manager	Date

CHANGE HISTORY LOG

Revision	Effective Date	Description of Changes
1	4/12/04	Created document from GCFE ver 9A flight VHDL Code

Table of Contents

Table of Contents 4
List of Figures 4
1 PURPOSE 5
2 DEFINITIONS 5
2.1 Acronyms 5
3 APPLICABLE DOCUMENTS 5
4 GCFE OVERVIEW 5
5 GCFE9A ASIC MODULES 7
5.1 GCFE-CHIP.VHD 7
5.2 GCFE_REG.VHD 14
5.3 GCFE_PKG.VHD 18
5.4 CMD_SEQ.VHD 23
5.5 READ_BACK.VHD 28
5.6 CTRL_ACQ.VHD 32
5.7 RANGE_CTRL.VHD 36
5.8 DIG_MUX.VHDL 44

List of Figures

Figure 1. GCFE Block Diagram 6

1 PURPOSE

This document documents the VHDL code in the GLAST Large Area Telescope (LAT) flight Calorimeter Front End version 9A (GCFE9A) ASIC.

2 DEFINITIONS

2.1 Acronyms

GLAST – Gamma-ray Large Area Space Telescope

GCFE – Glast Calorimeter Front-end Electronics

GCRC – Glast Calorimeter Readout Control electronics

LAT – Large Area Telescope

CAL – Calorimeter Detector

TREQ – Level 1 Trigger Request

TEM – Tower Electronics Module

3 APPLICABLE DOCUMENTS

[1] Calorimeter Readout Control ASIC – Conceptual Design, Submission 5, LAT-SS-00208-D4.

4 GCFE OVERVIEW

One of the two custom ASICs required for the GLAST Calorimeter is the Glast Calorimeter Front End (GCFE) Application Specific Integrated Circuit (ASIC). The basic function of the GCFE is to amplify and shape signals from 2 PIN photodiodes per crystal end and capture them in a track and hold circuit for digitization by an external ADC. Discriminators within the ASIC are provided externally for inputs to the GLAST trigger system. The digital control blocks within the GCFE handle command and data readout of configuration registers, control the data acquisition timing and auto ranging functions.

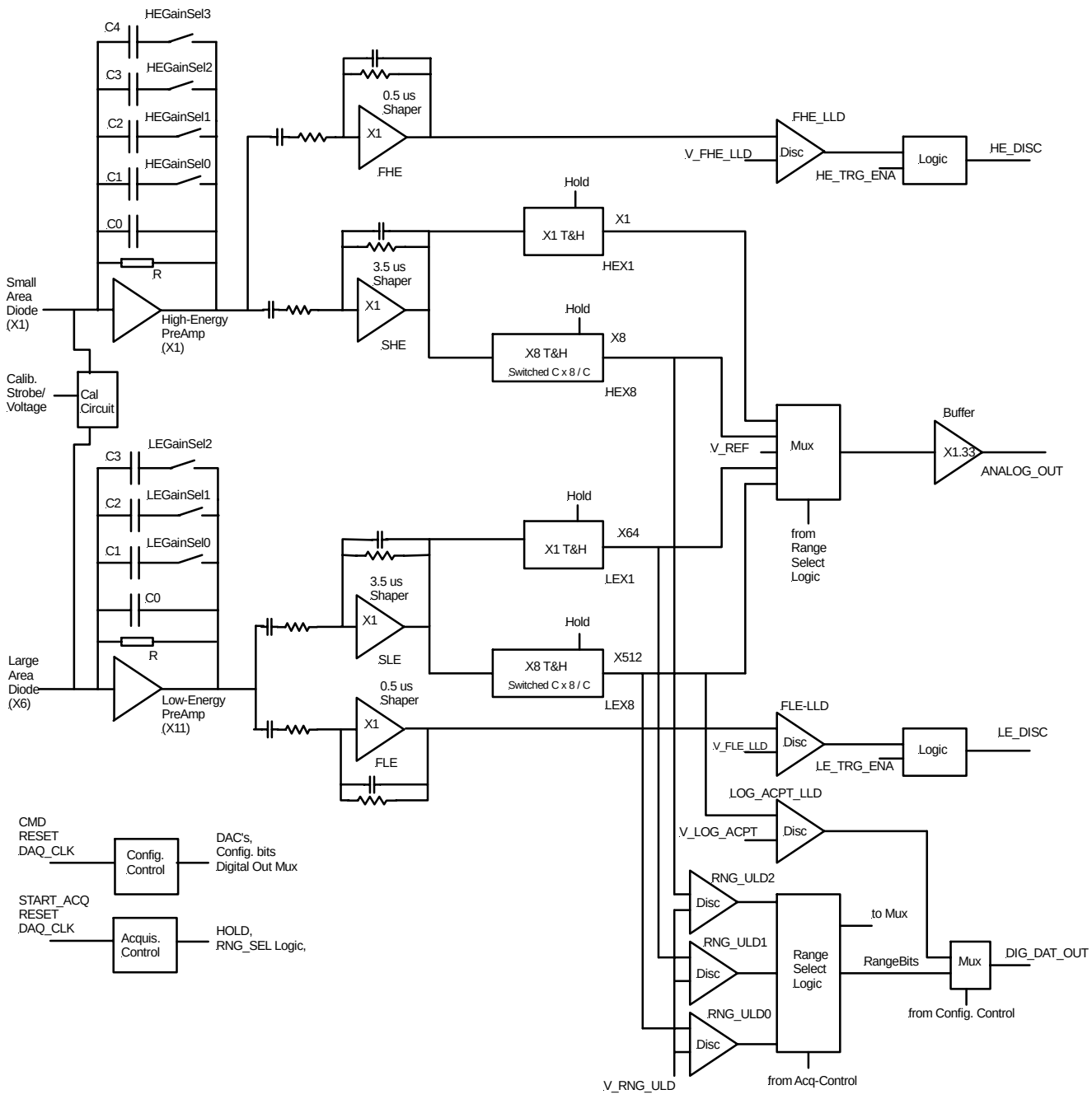


Figure 1. GCFE Block Diagram

5 GCFE9A ASIC MODULES

5.1 GCFE-CHIP.VHD

```
--
-- Created by David Nelson 1/24/01
--
-- Filename: gcfe_chip.vhd
--
-- Function: GCFE Top Level
--
-- This file describes all the signal interconnections between the various
-- components. Each signal had to be seperately called out instead of an
-- array brause the synthsis tools translated the names to internal "nxx"
-- names which makes it difficult to trace.
--
-- Modification:
--
--
--

--Library gcfe_vhdl;
Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use gcfe_pkg.all;


Entity GCFE_Chip is
Port (
--
-- Chip Input Signals From GCRC

-- Chip Address
signal ADDR0          : in std_logic; -- Chip Address
signal ADDR1          : in std_logic; -- Chip Address
signal ADDR2          : in std_logic; -- Chip Address
signal ADDR3          : in std_logic; -- Chip Address


signal RIGHT_FIRST     : in std_logic;       --

signal DAQ_CLK       : in std_logic;        --  Clock input from GCRC
signal CMD           : in std_logic;        --  Command/Data input from GCRC
signal RESET         : in std_logic;       --  RESET

signal START_ACQ     : in std_logic;        --  Start acquisition from GCRC

--
-- Chip Output Signals to the GCRC
--
Signal DIG_DAT_OUT     : buffer std_logic;   -- Data


signal HOLD             : buffer std_logic;    -- Hold to sample and hold circuits

signal RANGE_SEL0       : buffer std_logic;
signal RANGE_SEL1       : buffer std_logic;
```

```
signal RANGE_SEL2       : buffer std_logic;
signal RANGE_SEL3       : buffer std_logic;
signal RANGE_SEL4       : buffer std_logic;

signal RNG_ULD0         : in std_logic;
signal RNG_ULD1         : in std_logic;
signal RNG_ULD2         : in std_logic;
signal LOG_ACPT_LLD     : in std_logic;

signal LE_GN_SEL0        : buffer std_logic;
signal LE_GN_SEL1        : buffer std_logic;
signal LE_GN_SEL2        : buffer std_logic;

signal HE_GN_SEL0        : buffer std_logic;
signal HE_GN_SEL1        : buffer std_logic;
signal HE_GN_SEL2        : buffer std_logic;
signal HE_GN_SEL3        : buffer std_logic;

signal CALIB_GAIN       : buffer std_logic;
signal CALIB_HE_EN      : buffer std_logic;
signal CALIB_LE_EN      : buffer std_logic;

Signal  HE_TRG_ENA      : buffer std_logic;
Signal  LE_TRG_ENA      : buffer std_logic;
Signal  LOG_ACPT_ENA    : buffer std_logic;

signal FLE_DAC0          : buffer std_logic;
signal FLE_DAC1          : buffer std_logic;
signal FLE_DAC2          : buffer std_logic;
signal FLE_DAC3          : buffer std_logic;
signal FLE_DAC4          : buffer std_logic;
signal FLE_DAC5          : buffer std_logic;
signal FLE_DAC6          : buffer std_logic;

signal FHE_DAC0          : buffer std_logic;
signal FHE_DAC1          : buffer std_logic;
signal FHE_DAC2          : buffer std_logic;
signal FHE_DAC3          : buffer std_logic;
signal FHE_DAC4          : buffer std_logic;
signal FHE_DAC5          : buffer std_logic;
signal FHE_DAC6          : buffer std_logic;

signal LOG_ACPT_DAC0     : buffer std_logic;
signal LOG_ACPT_DAC1     : buffer std_logic;
signal LOG_ACPT_DAC2     : buffer std_logic;
signal LOG_ACPT_DAC3     : buffer std_logic;
signal LOG_ACPT_DAC4     : buffer std_logic;
signal LOG_ACPT_DAC5     : buffer std_logic;
signal LOG_ACPT_DAC6     : buffer std_logic;

signal RNG_ULD_DAC0      : buffer std_logic;
signal RNG_ULD_DAC1      : buffer std_logic;
signal RNG_ULD_DAC2      : buffer std_logic;
signal RNG_ULD_DAC3      : buffer std_logic;
signal RNG_ULD_DAC4      : buffer std_logic;
signal RNG_ULD_DAC5      : buffer std_logic;
signal RNG_ULD_DAC6      : buffer std_logic;

signal REF_DAC0          : buffer std_logic;
```

```
signal REF_DAC1          : buffer std_logic;
signal REF_DAC2          : buffer std_logic;
signal REF_DAC3          : buffer std_logic;
signal REF_DAC4          : buffer std_logic;
signal REF_DAC5          : buffer std_logic;
signal REF_DAC6          : buffer std_logic
  );
End GCFE_Chip;

Architecture Chip of GCFE_Chip is

--
--  Inter connect signals
--
--
-- CMD_SEQ

signal Addr             : std_logic_vector(3 downto 0);

signal Chip_Func        : std_logic_vector(4 downto 0);
signal Func_Strb        : std_logic;
signal Data_Strb        : std_logic;
signal Data_Reg      : std_logic_vector(15 downto 0);

-- GCFE_REG

Signal  FIRST_RNG       : std_logic_vector(1 downto 0);
Signal  USE_FRST_RNG    : std_logic;
Signal  OVERWRITE       : std_logic;
Signal  CTRL_SEL        : std_logic;
Signal  ENA_RNG         : std_logic_vector(1 downto 0);

-- GCFE_REG to GCFE_READ_BACK

signal  CTRL_REG_0      : std_logic_vector(15 downto 0);
signal  CTRL_REG_1      : std_logic_vector(15 downto 0);
signal  FLE_DAC_REG     : std_logic_vector(15 downto 0);
signal  FHE_DAC_REG     : std_logic_vector(15 downto 0);
signal  LOG_ACPT_DAC_REG  : std_logic_vector(15 downto 0);
signal  RNG_ULD_DAC_REG   : std_logic_vector(15 downto 0);
signal  REF_DAC_REG     : std_logic_vector(15 downto 0);
signal  CTRL_DATA       : std_logic;

-- RANGE_SEL;

Signal  RANGE_VOLT      : std_logic_vector( 1 downto 0);

-- DIG_MUX;


-- CTRL_ACQ;

signal SEL_LOG_ACPT     : std_logic;
Signal RNG_SEL_1        : std_logic;
Signal RNG_SEL_0        : std_logic;
Signal INC_RNG          : std_logic;
signal LD_RNG_ULD       : std_logic;

signal RANGE_SEL        : std_logic_vector(4 downto 0);
```

```
signal RNG_ULD          : std_logic_vector(2 downto 0);
signal LE_GN_SEL        : std_logic_vector(2 downto 0);
signal HE_GN_SEL        : std_logic_vector(3 downto 0);

signal FLE_DAC          : std_logic_vector(6 downto 0);
signal FHE_DAC          : std_logic_vector(6 downto 0);
signal LOG_ACPT_DAC     : std_logic_vector(6 downto 0);
signal RNG_ULD_DAC      : std_logic_vector(6 downto 0);
signal REF_DAC          : std_logic_vector(6 downto 0);

Begin

Change_Array1: process( RANGE_SEL,RNG_ULD0,RNG_ULD1,RNG_ULD2,LE_GN_SEL, HE_GN_SEL,
Addr0, Addr1, Addr2, Addr3)
begin

 RANGE_SEL0   <=   RANGE_SEL(0);
 RANGE_SEL1   <=   RANGE_SEL(1);
 RANGE_SEL2   <=   RANGE_SEL(2);
 RANGE_SEL3   <=   RANGE_SEL(3);
 RANGE_SEL4   <=   RANGE_SEL(4);

 RNG_ULD(0)     <=    RNG_ULD0;
 RNG_ULD(1)     <=    RNG_ULD1;
 RNG_ULD(2)     <=    RNG_ULD2;

 LE_GN_SEL0     <=    LE_GN_SEL(0);
 LE_GN_SEL1     <=    LE_GN_SEL(1);
 LE_GN_SEL2     <=    LE_GN_SEL(2);

 HE_GN_SEL0      <=    HE_GN_SEL(0);
 HE_GN_SEL1      <=    HE_GN_SEL(1);
 HE_GN_SEL2      <=    HE_GN_SEL(2);
 HE_GN_SEL3      <=    HE_GN_SEL(3);

 Addr(0)  <=    Addr0;
 Addr(1)  <=    Addr1;
 Addr(2)  <=    Addr2;
 Addr(3)  <=    Addr3;

end process;

Change_Array2: process( FLE_DAC, FHE_DAC,LOG_ACPT_DAC, RNG_ULD_DAC, REF_DAC)
begin

 FLE_DAC0 <=    FLE_DAC(0);
 FLE_DAC1 <=    FLE_DAC(1);
 FLE_DAC2 <=    FLE_DAC(2);
 FLE_DAC3 <=    FLE_DAC(3);
 FLE_DAC4 <=    FLE_DAC(4);
 FLE_DAC5 <=    FLE_DAC(5);
 FLE_DAC6 <=    FLE_DAC(6);

 FHE_DAC0 <=    FHE_DAC(0);
 FHE_DAC1 <=    FHE_DAC(1);
 FHE_DAC2 <=    FHE_DAC(2);
 FHE_DAC3 <=    FHE_DAC(3);
 FHE_DAC4 <=    FHE_DAC(4);
 FHE_DAC5 <=    FHE_DAC(5);
```

```
 FHE_DAC6 <=    FHE_DAC(6);

 LOG_ACPT_DAC0 <=    LOG_ACPT_DAC(0);
 LOG_ACPT_DAC1 <=    LOG_ACPT_DAC(1);
 LOG_ACPT_DAC2 <=    LOG_ACPT_DAC(2);
 LOG_ACPT_DAC3 <=    LOG_ACPT_DAC(3);
 LOG_ACPT_DAC4 <=    LOG_ACPT_DAC(4);
 LOG_ACPT_DAC5 <=    LOG_ACPT_DAC(5);
 LOG_ACPT_DAC6 <=    LOG_ACPT_DAC(6);

 RNG_ULD_DAC0 <=    RNG_ULD_DAC(0);
 RNG_ULD_DAC1 <=    RNG_ULD_DAC(1);
 RNG_ULD_DAC2 <=    RNG_ULD_DAC(2);
 RNG_ULD_DAC3 <=    RNG_ULD_DAC(3);
 RNG_ULD_DAC4 <=    RNG_ULD_DAC(4);
 RNG_ULD_DAC5 <=    RNG_ULD_DAC(5);
 RNG_ULD_DAC6 <=    RNG_ULD_DAC(6);

 REF_DAC0 <=    REF_DAC(0);
 REF_DAC1 <=    REF_DAC(1);
 REF_DAC2 <=    REF_DAC(2);
 REF_DAC3 <=    REF_DAC(3);
 REF_DAC4 <=    REF_DAC(4);
 REF_DAC5 <=    REF_DAC(5);
 REF_DAC6 <=    REF_DAC(6);

end process; -- Change_Array

U_CMD_SEQ: CMD_SEQ
port map (
    DAQ_CLK         => DAQ_CLK,
    CMD             => CMD,
    ADDR            => ADDR,
    RIGHT_FIRST     => RIGHT_FIRST,
    Chip_Func       => Chip_Func,
    Func_Strb       => Func_Strb,
    Data_Strb       => Data_Strb,
    Data_Reg        => Data_Reg,
    RESET           => RESET
    ); -- CMD_SEQ

U_GCFE_REG: GCFE_REG
port map (
    DAQ_CLK         => DAQ_CLK,
    Data_Reg        => Data_Reg,
    Chip_Func       => Chip_Func,
    Func_strb       => Func_strb,
    Data_Strb       => Data_Strb,
    CTRL_REG_0      => CTRL_REG_0,
    CTRL_REG_1      => CTRL_REG_1,
    FLE_DAC_REG     => FLE_DAC_REG,
    FHE_DAC_REG     => FHE_DAC_REG,
    LOG_ACPT_DAC_REG => LOG_ACPT_DAC_REG,
    RNG_ULD_DAC_REG  => RNG_ULD_DAC_REG,
    REF_DAC_REG     => REF_DAC_REG,
    FHE_DAC         => FHE_DAC,
    FLE_DAC         => FLE_DAC,
    LOG_ACPT_DAC    => LOG_ACPT_DAC,
    RNG_ULD_DAC     => RNG_ULD_DAC,
```

```
    USE_FRST_RNG    => USE_FRST_RNG,
    OVERWRITE       => OVERWRITE,
    REF_DAC         => REF_DAC,
    LE_GN_SEL       => LE_GN_SEL,
    HE_GN_SEL       => HE_GN_SEL,
    FIRST_RNG       => FIRST_RNG,
    ENA_RNG         => ENA_RNG,
    LOG_ACPT_ENA    => LOG_ACPT_ENA,
    LE_TRG_ENA      => LE_TRG_ENA,
    HE_TRG_ENA      => HE_TRG_ENA,
    CALIB_GAIN      => CALIB_GAIN,
    CALIB_LE_EN     => CALIB_LE_EN,
    CALIB_HE_EN     => CALIB_HE_EN
    );

U_CTRL_ACQ: CTRL_ACQ
Port map (
    DAQ_CLK      => DAQ_CLK,
    START_ACQ    => START_ACQ,
   RESET       => RESET,
    INC_RNG      => INC_RNG,
    LD_RNG_ULD      => LD_RNG_ULD,
    HOLD            => HOLD,
    SEL_LOG_ACPT    => SEL_LOG_ACPT,
   RNG_SEL_1      => RNG_SEL_1,
    RNG_SEL_0    => RNG_SEL_0
   ); -- CTRL_ACQ

U_RANGE_CNTRL : RANGE_CNTRL
Port map (
    DAQ_CLK      => DAQ_CLK,
    RESET      => RESET,
   USE_FRST_RNG    => USE_FRST_RNG,
   OVERWRITE   => OVERWRITE,
   RNG_ULD        => RNG_ULD,
    FIRST_RNG       => FIRST_RNG,
    ENA_RNG         => ENA_RNG,
   INC_RNG     => INC_RNG,
    RANGE_SEL       => RANGE_SEL,
    LD_RNG_ULD      => LD_RNG_ULD,
    RANGE_VOLT      => RANGE_VOLT
   ); -- RANGE_CNTRL

U_READ_BACK : READ_BACK
port map (
    DAQ_CLK        => DAQ_CLK,
    RESET           => RESET,
    Func_Strb      => Func_Strb,
    Chip_Func      => Chip_Func,
    CTRL_REG_0     => CTRL_REG_0,
    CTRL_REG_1      => CTRL_REG_1,
    FLE_DAC_REG     => FLE_DAC_REG,
    FHE_DAC_REG     => FHE_DAC_REG,
    LOG_ACPT_DAC_REG => LOG_ACPT_DAC_REG,
    RNG_ULD_DAC_REG => RNG_ULD_DAC_REG,
    REF_DAC_REG     => REF_DAC_REG,
    CTRL_DATA       => CTRL_DATA,
    CTRL_SEL        => CTRL_SEL
    ); -- READ_BACK
```

```
U_DIG_MUX: DIG_MUX
Port map (
    LOG_ACPT_LLD    => LOG_ACPT_LLD,
    SEL_LOG_ACPT    => SEL_LOG_ACPT,
    CTRL_SEL     => CTRL_SEL,
    CTRL_DATA       => CTRL_DATA,
    RNG_SEL_1       => RNG_SEL_1,
    RNG_SEL_0       => RNG_SEL_0,
   RANGE_VOLT     => RANGE_VOLT,
   DIG_DAT_OUT    => DIG_DAT_OUT
   ); -- DIG_MUX

End Chip;
```

5.2 GCFE_REG.VHD

```
--
-- Created by David Nelson Feb 8, 2001
--
--  Filename: gcfe_reg.vhd
--
--  Function: Load and Read all of the GCFE registers listed in Table 2, section 7
--  of the GCFE ASIC specification. This code uses the Chip function, Func_Strb, and
Data_Strb
--  from CMD_SEQ code.
--
--  Modifications:
--
--
--Library gcfe_vhdl;
Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use gcfe_pkg.all;

Entity GCFE_Reg is
Port   (
  DAQ_CLK     : in std_logic;
  Chip_Func      : in std_logic_vector(4 downto 0);
  Func_Strb   : in std_logic;
  Data_Strb   : in std_logic;
    Data_Reg        : in std_logic_vector(15 downto 0);
    USE_FRST_RNG    : buffer std_logic;
    FLE_DAC         : buffer std_logic_vector(6 downto 0);
    FHE_DAC         : buffer std_logic_vector(6 downto 0);
    LOG_ACPT_DAC    : buffer std_logic_vector(6 downto 0);
    RNG_ULD_DAC     : buffer std_logic_vector(6 downto 0);
    REF_DAC         : buffer std_logic_vector(6 downto 0);
  CTRL_REG_0    : buffer std_logic_vector(15 downto 0);
    CTRL_REG_1      : buffer std_logic_vector(15 downto 0);
    FLE_DAC_REG     : buffer std_logic_vector(15 downto 0);
    FHE_DAC_REG     : buffer std_logic_vector(15 downto 0);
    LOG_ACPT_DAC_REG    : buffer std_logic_vector(15 downto 0);
    RNG_ULD_DAC_REG     : buffer std_logic_vector(15 downto 0);
    REF_DAC_REG         : buffer std_logic_vector(15 downto 0);
    LE_GN_SEL       : buffer std_logic_vector(2 downto 0);
    HE_GN_SEL       : buffer std_logic_vector(3 downto 0);
    ENA_RNG         : buffer std_logic_vector(1 downto 0);
    LOG_ACPT_ENA    : buffer std_logic;
    LE_TRG_ENA      : buffer std_logic;
    HE_TRG_ENA      : buffer std_logic;
    CALIB_GAIN      : buffer std_logic;
    CALIB_LE_EN     : buffer std_logic;
    CALIB_HE_EN     : buffer std_logic;
    FIRST_RNG       : buffer std_logic_vector( 1 downto 0);
    OVERWRITE       : buffer std_logic
  );
 End GCFE_Reg;

Architecture Behaviour of GCFE_Reg is

Begin
```

```
-- This process loads the CONFIG_REG_0 listed in Table 2,4 of the
-- GCFE ASIC specification

Ctrl_Reg2: Process(DAQ_CLK, Data_Strb, Data_reg, Chip_Func,CTRL_REG_0,
                     LE_GN_SEL, HE_GN_SEL,FIRST_RNG )
Begin
   if (DAQ_CLK'event and DAQ_CLK = '1') then

  if (Data_Strb = '1' and Chip_Func = LD_CONFIG_REG_0_CMD) then
        CTRL_REG_0              <= Data_Reg;
        LE_GN_SEL(2 downto 0)   <= Data_Reg(2 downto 0);
        HE_GN_SEL(3 downto 0)   <= Data_Reg(6 downto 3);
        ENA_RNG(0)              <= Data_Reg(7);
        ENA_RNG(1)              <= Data_Reg(8);
        USE_FRST_RNG            <= Data_Reg(9);
        FIRST_RNG(1 downto 0)   <= Data_Reg(11 downto 10);
        OVERWRITE               <= Data_Reg(12);
  else
     CTRL_REG_0(15 downto 0) <= CTRL_REG_0(15 downto 0);
        LE_GN_SEL(2 downto 0)   <= LE_GN_SEL(2 downto 0);
        HE_GN_SEL(3 downto 0)   <= HE_GN_SEL(3 downto 0);
        ENA_RNG                 <= ENA_RNG;
        USE_FRST_RNG            <= USE_FRST_RNG;
        FIRST_RNG(1 downto 0)   <= FIRST_RNG(1 downto 0);
        OVERWRITE               <= OVERWRITE;
  end if;
    else

  CTRL_REG_0(15 downto 0) <= CTRL_REG_0(15 downto 0);
        LE_GN_SEL(2 downto 0)   <= LE_GN_SEL(2 downto 0);
        HE_GN_SEL(3 downto 0)   <= HE_GN_SEL(3 downto 0);
        ENA_RNG                 <= ENA_RNG;
        USE_FRST_RNG            <= USE_FRST_RNG;
        FIRST_RNG(1 downto 0)   <= FIRST_RNG(1 downto 0);
        OVERWRITE               <= OVERWRITE;
   end if;
end process;

-- This process loads the CONFIG_REG_1 listed in Table 2,3 of the
--  GCFE ASIC specification

Ctrl_Reg3: Process(DAQ_CLK, Data_Strb, Data_reg, Chip_Func,CTRL_REG_1)
Begin
   if (DAQ_CLK'event and DAQ_CLK = '1') then

    if (Data_Strb = '1' and Chip_Func = LD_CONFIG_REG_1_CMD) then
        CTRL_REG_1              <= Data_Reg;
        LOG_ACPT_ENA            <= Data_Reg(0);
        LE_TRG_ENA              <= Data_Reg(1);
        HE_TRG_ENA              <= Data_Reg(2);
        CALIB_GAIN              <= Data_Reg(3);
        CALIB_LE_EN             <= Data_Reg(4);
        CALIB_HE_EN             <= Data_Reg(5);
    else
     CTRL_REG_1(15 downto 0) <= CTRL_REG_1(15 downto 0);
        LOG_ACPT_ENA            <= LOG_ACPT_ENA;
        LE_TRG_ENA              <= LE_TRG_ENA;
        HE_TRG_ENA              <= HE_TRG_ENA;
```

```
        CALIB_GAIN              <= CALIB_GAIN;
        CALIB_LE_EN             <= CALIB_LE_EN;
        CALIB_HE_EN             <= CALIB_HE_EN;

    end if;
   else
  CTRL_REG_1(15 downto 0) <= CTRL_REG_1(15 downto 0);
        LOG_ACPT_ENA            <= LOG_ACPT_ENA;
        LE_TRG_ENA              <= LE_TRG_ENA;
        HE_TRG_ENA              <= HE_TRG_ENA;
        CALIB_GAIN              <= CALIB_GAIN;
        CALIB_LE_EN             <= CALIB_LE_EN;
        CALIB_HE_EN             <= CALIB_HE_EN;
   end if;
end process;

-- This process loads the FLE_DAC listed in Table 2 of the
-- GCFE ASIC specification

Ctrl_Reg4: Process(DAQ_CLK, Data_Strb, Data_reg, Chip_Func,FLE_DAC_REG)
Begin
   if (DAQ_CLK'event and DAQ_CLK = '1') then

    if (Data_Strb = '1' and Chip_Func = LD_FLE_DAC_CMD) then
        FLE_DAC_REG                 <= Data_Reg(15 downto 0);
        FLE_DAC                     <= Data_Reg(6 downto 0);
    else
       FLE_DAC_REG(15 downto 0)    <= FLE_DAC_REG(15 downto 0);
        FLE_DAC                     <= FLE_DAC;
    end if;
   else
  FLE_DAC_REG(15 downto 0)    <= FLE_DAC_REG(15 downto 0);
        FLE_DAC                     <= FLE_DAC;
   end if;
end process;

-- This process loads the FHE_DAC listed in Table 2 of the
--  GCFE ASIC specification

Ctrl_Reg5: Process(DAQ_CLK, Data_Strb, Data_reg, Chip_Func,FHE_DAC_REG)
Begin
   if (DAQ_CLK'event and DAQ_CLK = '1') then

    if (Data_Strb = '1' and Chip_Func = LD_FHE_DAC_CMD) then
        FHE_DAC_REG                 <= Data_Reg(15 downto 0);
        FHE_DAC                     <= Data_Reg(6 downto 0);
    else
     FHE_DAC_REG(15 downto 0)    <= FHE_DAC_REG(15 downto 0);
        FHE_DAC                     <= FHE_DAC;
    end if;
    else
        FHE_DAC_REG(15 downto 0)    <= FHE_DAC_REG(15 downto 0);
        FHE_DAC                     <= FHE_DAC;
    end if;
end process;

-- This process loads the FHE_DAC listed in Table 2 of the
-- GCFE ASIC specification
```

```
Ctrl_Reg6: Process(DAQ_CLK, Data_Strb, Data_reg,
                     Chip_Func,LOG_ACPT_DAC_REG)
Begin
   if (DAQ_CLK'event and DAQ_CLK = '1') then

    if (Data_Strb = '1' and Chip_Func = LD_LOG_ACPT_DAC_CMD) then
        LOG_ACPT_DAC_REG            <= Data_Reg(15 downto 0);
        LOG_ACPT_DAC                <= Data_Reg(6 downto 0);
    else
     LOG_ACPT_DAC_REG(15 downto 0)   <= LOG_ACPT_DAC_REG(15 downto 0);
        LOG_ACPT_DAC                    <= LOG_ACPT_DAC;
    end if;
   else
  LOG_ACPT_DAC_REG(15 downto 0)   <= LOG_ACPT_DAC_REG(15 downto 0);
        LOG_ACPT_DAC                    <= LOG_ACPT_DAC;
   end if;
end process;

-- This process loads the RNG_ULD_DAC listed in Table 2 of the GCFE ASIC
-- specification

Ctrl_Reg7: Process(DAQ_CLK, Data_Strb, Data_reg, Chip_Func,RNG_ULD_DAC_REG)
Begin
   if (DAQ_CLK'event and DAQ_CLK = '1') then

    if (Data_Strb = '1' and Chip_Func = LD_RNG_ULD_DAC_CMD) then
        RNG_ULD_DAC_REG                 <= Data_Reg(15 downto 0);
        RNG_ULD_DAC                     <= Data_Reg(6 downto 0);
    else
     RNG_ULD_DAC_REG(15 downto 0)    <= RNG_ULD_DAC_REG(15 downto 0);
        RNG_ULD_DAC                     <= RNG_ULD_DAC;
    end if;
   else
  RNG_ULD_DAC_REG(15 downto 0)    <= RNG_ULD_DAC_REG(15 downto 0);
        RNG_ULD_DAC                     <= RNG_ULD_DAC;
   end if;
end process;

-- This process loads the REF_DAC listed in Table 2 of the
--  GCFE ASIC specification

Ctrl_Reg8: Process(DAQ_CLK, Data_Strb, Data_reg, Chip_Func,REF_DAC_REG)
Begin
   if (DAQ_CLK'event and DAQ_CLK = '1') then

    if (Data_Strb = '1' and Chip_Func = LD_REF_DAC_CMD) then
        REF_DAC_REG                 <= Data_Reg(15 downto 0);
        REF_DAC                     <= Data_Reg(6 downto 0);
    else
     REF_DAC_REG(15 downto 0)    <= REF_DAC_REG(15 downto 0);
        REF_DAC                     <= REF_DAC;
    end if;
  else
     REF_DAC_REG(15 downto 0)    <= REF_DAC_REG(15 downto 0);
        REF_DAC                     <= REF_DAC;
  end if;

end process;
End Behaviour;
```

5.3 GCFE_PKG.VHD

```
--
--
-- Created by David Nelson Jan 24, 2001
--
--  Filename: gcfe_pkg.vhd
--
--  Function: Component definition file for GCFE
--
--  These eight VHDL files are designed using the Conceptional Design of the Glast
--  Calorimeter Front-End Electronics (GCFE) ASIC, document #89. Refer to
--  Figure titled GCFE Block Diagram. The Config. Control
--  block is implemented with CMD_SEQ, GCFE_REG, and READ_BACK. The Acquis.
--  Control is implemented with CTRL_ACQ. The Range Select Logic is implemented
--  with RANGE_CNTRL. The Mux block is implemented with DIG_MUX.
--
--
--  There are 7 components defined in this file.
--  (1) GCFE_Chip, which defines all interface signals comming in and leaving
--the digital section.
--(2) CMD_SEQ, which decodes uses the DAQ_ClK to decode CMD and generate function
--  strobes and data strobes.
--  (3) GCFE_REG, which holds all of the setup values such as DACs and configuration
--   registers.
--  (4) RANGE_CNTRL, which looks at the configuration registers for mode control
--  and the three range discriminators and sets the proper range for the analog
--  output that is sent to the ADC.
--  (5) READ_BACK, which is used to read back all of the DAC settings and the two
--  configuration registers.
--  (6) DIG_MUX, which is use to multiplex the readback of configuration registers,
--  DACs and the range bits from the selected ADC range.
--  (7) CTRL_ACQ, which is used to generate analog readout sequence.
--
--  There is an eight file "gcfe_chip.vhl" that defines all the interconnecting
--  signals between the priviously
--  described blocks.
--
--  Modifications:
--
-- 3/14/2001
-- Removed rate counters

--Library gcfe_vhdl;
Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

Package gcfe_pkg is

--
-- Constant definitions for Function register in GCFE
--

-- Dataless Functions

Constant NOP_CMD                : std_logic_vector(4 downto 0) := "00000";
Constant RESET_CNTR_CMD        : std_logic_vector(4 downto 0) := "00001";
Constant STOP_CNTR_CMD         : std_logic_vector(4 downto 0) := "00010";
```

```
-- Load Functions

Constant LD_CONFIG_REG_0_CMD    : std_logic_vector(4 downto 0) := "00100";
Constant LD_CONFIG_REG_1_CMD    : std_logic_vector(4 downto 0) := "00101";
Constant LD_FLE_DAC_CMD       : std_logic_vector(4 downto 0) := "00110";
Constant LD_FHE_DAC_CMD       : std_logic_vector(4 downto 0) := "00111";
Constant LD_LOG_ACPT_DAC_CMD    : std_logic_vector(4 downto 0) := "01000";
Constant LD_RNG_ULD_DAC_CMD     : std_logic_vector(4 downto 0) := "01001";
Constant LD_REF_DAC_CMD       : std_logic_vector(4 downto 0) := "01010";

-- Read Functions

Constant RD_CONFIG_REG_0_CMD    : std_logic_vector(4 downto 0) := "10100";
Constant RD_CONFIG_REG_1_CMD    : std_logic_vector(4 downto 0) := "10101";
Constant RD_FLE_DAC_CMD       : std_logic_vector(4 downto 0) := "10110";
Constant RD_FHE_DAC_CMD       : std_logic_vector(4 downto 0) := "10111";
Constant RD_LOG_ACPT_DAC_CMD    : std_logic_vector(4 downto 0) := "11000";
Constant RD_RNG_ULD_DAC_CMD     : std_logic_vector(4 downto 0) := "11001";
Constant RD_REF_DAC_CMD       : std_logic_vector(4 downto 0) := "11010";

Component GCFE_Chip
Port (
-- Bonded Chip Address
    ADDR0           : in std_logic ;
    ADDR1           : in std_logic ;
    ADDR2           : in std_logic ;
    ADDR3           : in std_logic ;

  RIGHT_FIRST     : in std_logic;                   -- 1 => Right, 0 => Left
  DAQ_CLK       : in std_logic;                   --  Clock input from GCRC
  CMD             : in std_logic;                   --  Command/Data input from GCRC
  RESET         : in std_logic;                 --  RESET
  START_ACQ     : in std_logic;                 --  Start acq
--
-- Chip Output Signals to the GCRC
--
  DIG_DAT_OUT     : buffer std_logic;                 -- Data
--
--  Signals that are external to this digital design but inside the gcfe
--
    HOLD            : buffer std_logic;
    RANGE_SEL0     : buffer std_logic;
    RANGE_SEL1      : buffer std_logic;
    RANGE_SEL2      : buffer std_logic;
    RANGE_SEL3      : buffer std_logic;
    RANGE_SEL4      : buffer std_logic;

    RNG_ULD0         : in std_logic;
    RNG_ULD1         : in std_logic;
    RNG_ULD2         : in std_logic;

    LOG_ACPT_LLD    : in std_logic;

    LE_GN_SEL0       : buffer std_logic;
    LE_GN_SEL1       : buffer std_logic;
    LE_GN_SEL2       : buffer std_logic;

    HE_GN_SEL0       : buffer std_logic;
```

```
    HE_GN_SEL1       : buffer std_logic;
    HE_GN_SEL2       : buffer std_logic;
    HE_GN_SEL3       : buffer std_logic;

    LOG_ACPT_ENA    : buffer std_logic;
    LE_TRG_ENA      : buffer std_logic;
    HE_TRG_ENA      : buffer std_logic;

    CALIB_GAIN      : buffer std_logic;
    CALIB_HE_EN     : buffer std_logic;
    CALIB_LE_EN     : buffer std_logic;


    FLE_DAC0          : buffer std_logic;
    FLE_DAC1          : buffer std_logic;
    FLE_DAC2          : buffer std_logic;
    FLE_DAC3          : buffer std_logic;
    FLE_DAC4          : buffer std_logic;
    FLE_DAC5          : buffer std_logic;
    FLE_DAC6          : buffer std_logic;

    FHE_DAC0          : buffer std_logic;
    FHE_DAC1          : buffer std_logic;
    FHE_DAC2          : buffer std_logic;
    FHE_DAC3          : buffer std_logic;
    FHE_DAC4          : buffer std_logic;
    FHE_DAC5          : buffer std_logic;
    FHE_DAC6          : buffer std_logic;

    LOG_ACPT_DAC0     : buffer std_logic;
    LOG_ACPT_DAC1     : buffer std_logic;
    LOG_ACPT_DAC2     : buffer std_logic;
    LOG_ACPT_DAC3     : buffer std_logic;
    LOG_ACPT_DAC4     : buffer std_logic;
    LOG_ACPT_DAC5     : buffer std_logic;
    LOG_ACPT_DAC6     : buffer std_logic;


    RNG_ULD_DAC0      : buffer std_logic;
    RNG_ULD_DAC1      : buffer std_logic;
    RNG_ULD_DAC2      : buffer std_logic;
    RNG_ULD_DAC3      : buffer std_logic;
    RNG_ULD_DAC4      : buffer std_logic;
    RNG_ULD_DAC5      : buffer std_logic;
    RNG_ULD_DAC6      : buffer std_logic;


    REF_DAC0          : buffer std_logic;
    REF_DAC1          : buffer std_logic;
    REF_DAC2          : buffer std_logic;
    REF_DAC3          : buffer std_logic;
    REF_DAC4          : buffer std_logic;
    REF_DAC5          : buffer std_logic;
    REF_DAC6          : buffer std_logic
  );
End Component; -- GCFE_Chip

component CMD_SEQ
port (
```

```
    DAQ_CLK         : in std_logic;
    CMD             : in std_logic;
    ADDR            : in std_logic_vector(3 downto 0);
    RIGHT_FIRST     : in std_logic;
    Chip_Func       : buffer std_logic_vector(4 downto 0);
    Func_Strb       : buffer std_logic;
    Data_Strb       : buffer std_logic;
    Data_Reg        : buffer std_logic_vector(15 downto 0);
    RESET           : in std_logic
    );
end component;   -- CMD_SEQ

component GCFE_REG
port (
    DAQ_CLK         : in std_logic;
    Data_Reg        : in std_logic_vector(15 downto 0);
    Chip_Func       : in std_logic_vector(4 downto 0);
    Func_strb       : in std_logic;
    Data_Strb       : in std_logic;
    USE_FRST_RNG    : buffer std_logic;
    OVERWRITE       : buffer std_logic;
    CTRL_REG_0      : buffer std_logic_vector(15 downto 0);
    CTRL_REG_1      : buffer std_logic_vector(15 downto 0);
    FLE_DAC_REG     : buffer std_logic_vector(15 downto 0);
    FHE_DAC_REG     : buffer std_logic_vector(15 downto 0);
    REF_DAC_REG     : buffer std_logic_vector(15 downto 0);
    LOG_ACPT_DAC_REG    : buffer std_logic_vector(15 downto 0);
    RNG_ULD_DAC_REG     : buffer std_logic_vector(15 downto 0);
    FHE_DAC         : buffer std_logic_vector(6 downto 0);
    FLE_DAC         : buffer std_logic_vector(6 downto 0);
    LOG_ACPT_DAC    : buffer std_logic_vector(6 downto 0);
    RNG_ULD_DAC     : buffer std_logic_vector(6 downto 0);
    REF_DAC         : buffer std_logic_vector(6 downto 0);
    LE_GN_SEL       : buffer std_logic_vector(2 downto 0);
    HE_GN_SEL       : buffer std_logic_vector(3 downto 0);
    FIRST_RNG       : buffer std_logic_vector(1 downto 0);
    ENA_RNG         : buffer std_logic_vector(1 downto 0);
    CALIB_GAIN      : buffer std_logic;
    CALIB_HE_EN     : buffer std_logic;
    CALIB_LE_EN     : buffer std_logic;
    HE_TRG_ENA      : buffer std_logic;
    LE_TRG_ENA      : buffer std_logic;
    LOG_ACPT_ENA    : buffer std_logic
    );
 End Component; -- GCFE_REG

Component RANGE_CNTRL
Port   (
    DAQ_CLK         : in std_logic;
    RESET      : in std_logic;
   USE_FRST_RNG   : in std_logic;
   OVERWRITE   : in std_logic;

   RNG_ULD        : in std_logic_vector(2 downto 0);
    FIRST_RNG       : in std_logic_vector(1 downto 0);
    ENA_RNG         : in std_logic_vector(1 downto 0);
   INC_RNG     : in std_logic;
    LD_RNG_ULD      : in std_logic;
    RANGE_SEL       : buffer std_logic_vector(4 downto 0);
```

```
    RANGE_VOLT      : out std_logic_vector(1 downto 0)
   );
End Component; --RANGE_CNTRL;

component READ_BACK
port (
   DAQ_CLK     : in std_logic;
   RESET       : in std_logic;
   Func_Strb   : in std_logic;
   Chip_Func   : in std_logic_vector(4 downto 0);
    CTRL_REG_0   : in std_logic_vector(15 downto 0);
    CTRL_REG_1      : in std_logic_vector(15 downto 0);
    FLE_DAC_REG     : in std_logic_vector(15 downto 0);
    FHE_DAC_REG     : in std_logic_vector(15 downto 0);
    LOG_ACPT_DAC_REG : in std_logic_vector(15 downto 0);
    RNG_ULD_DAC_REG  : in std_logic_vector(15 downto 0);
    REF_DAC_REG     : in std_logic_vector(15 downto 0);
   CTRL_DATA      : out std_logic;
    CTRL_SEL        : out std_logic
    );
end component; -- READ_BACK

Component DIG_MUX
Port (
   LOG_ACPT_LLD     : in std_logic;
   CTRL_SEL    : in std_logic;
    CTRL_DATA       : in std_logic;
    RNG_SEL_1       : in std_logic;
    RNG_SEL_0       : in std_logic;
    SEL_LOG_ACPT    : in std_logic;
   RANGE_VOLT     : in std_logic_vector(1 downto 0);
   DIG_DAT_OUT    : buffer std_logic
   );
End Component; -- DIG_MUX

component CTRL_ACQ
port (
    DAQ_CLK         : in std_logic;
    START_ACQ       : in std_logic;
    RESET           : in std_logic;
    INC_RNG         : out std_logic;
    LD_RNG_ULD      : out std_logic;
    HOLD            : buffer std_logic;
    SEL_LOG_ACPT    : out std_logic;
    RNG_SEL_1       : out std_logic;
    RNG_SEL_0       : out std_logic
    );
end component; -- CTRL_ACQ



End gcfe_pkg; -- Package
```

5.4 CMD_SEQ.VHD

```
--
--
--
-- Created by David Nelson Jan 24, 2001
--
--
--  Function: Component definition file for GCFE
--
--  This code along with "GCFE_REG.VHD", "READ_BACK.VHD" implements section
--  7 of the ASIC specification which describes all of the commands and
-- read back functions.

--  What this code does....
--  Wait for start bit
--  then Shift CMD using the DAQ_CLK and get the command and address
--  from the input.
--  Check address to see if it matches Bonded Chip Address.
--  The code generates a command strobe and a data strobe if this chip
--  is addressed.
--  Wait until cmd_seq process completes, then wait for start bit again.
--
--  Filename: cmd_seq.vhd
--
--  Function:
--  Modifications:
--
--

--Library gcfe_vhdl;
Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use gcfe_pkg.all;


Entity CMD_Seq is
Port   (

-- Input signals from GCRC

  DAQ_CLK     : in      std_logic;
  CMD           : in    std_logic;

-- Bonded Chip Address
  ADDR          : in    std_logic_vector(3 downto 0);
    RIGHT_FIRST     : in        std_logic;

-- Outputs to other processes

  Chip_Func   : buffer std_logic_vector(4 downto 0);
  Func_Strb   : buffer std_logic;
  Data_Strb   : buffer std_logic;
    Data_Reg        : buffer std_logic_vector(15 downto 0);

-- Input from other processes
  RESET       : in      std_logic
  );
```

```
End CMD_Seq;


Architecture Behaviour of CMD_Seq is

signal Data_Shift_Reg   : std_logic_vector(15 downto 0);
signal Cmd_Shift_Reg    : std_logic_vector(8 downto 0);
signal GCFE_ADDR   : std_logic_vector(3 downto 0);
Signal Ld_CMD_Reg     : std_logic;
Signal Ld_DAT_Reg     : std_logic;
Signal Count       : std_logic_vector(4 downto 0);
Signal Next_Count     : std_logic_vector(4 downto 0);
Signal Ld_Count       : std_logic;
Signal Dn_Count       : std_logic;
signal ADDR_LR        : std_logic_vector(3 downto 0);

Type CMD_State is
  (
  Idle,
  CMD_In,
  DAT_In,
    Term
  );

Signal Present_State, Next_State  : CMD_State;


Begin

Cntr: Process(DAQ_CLK, RESET)
Begin
if (RESET = '1') then
    Count <= "00000";
elsif (DAQ_CLK'event and DAQ_CLK = '1') then
  if (Ld_Count = '1') then
     Count <= Next_Count;
  elsif (Dn_Count = '1') then
     Count <= Count - 1;
  else
     Count <= Count;
  end if;
end if;
End Process;  -- Cntr

-- CMD_SM Sequences through the Command State Machine

CMD_SM: Process(RESET, DAQ_CLK)
    Begin
     if (RESET = '1') then
        Present_State <= Idle;
     elsif (DAQ_CLK'event and DAQ_CLK = '1') then
         if (RESET = '1') then
         Present_State <= Idle;
     else
     Present_State <= Next_State;
        end if;
    end if;
    End Process; -- CMD_SM
```

```
-- Load_CMD transfers data from the input shift register to
-- Static register for other processes

Load_CMD: Process(DAQ_CLK, RIGHT_FIRST, ADDR, RESET)

Begin

if (RIGHT_FIRST = '0') then  -- This part of the code translates the address pins
    ADDR_LR <= ADDR;         -- if the chip is programmed to be a right chip
else                         -- This is decribed in section 10 of the ASIC
specification
    case ADDR is
    when "0000" => ADDR_LR <= "1011";
    when "0001" => ADDR_LR <= "1010";
    when "0010" => ADDR_LR <= "1001";
    when "0011" => ADDR_LR <= "1000";
    when "0100" => ADDR_LR <= "0111";
    when "0101" => ADDR_LR <= "0110";
    when "0110" => ADDR_LR <= "0101";
    when "0111" => ADDR_LR <= "0100";
    when "1000" => ADDR_LR <= "0011";
    when "1001" => ADDR_LR <= "0010";
    when "1010" => ADDR_LR <= "0001";
    when "1011" => ADDR_LR <= "0000";
    when others => ADDR_LR <= "0000";
    end case;
end if;

if (RESET = '1') then
     GCFE_ADDR      <= "0000";
     Chip_FUNC      <= "00000";
     Func_Strb        <= '0';
elsif (DAQ_CLK'event and DAQ_CLK = '1') then

   if (Ld_CMD_Reg = '1' ) then
     GCFE_ADDR <= Cmd_Shift_Reg(8 downto 5);
     Chip_FUNC <= Cmd_Shift_Reg(4 downto 0);

     if (Cmd_Shift_Reg(8 downto 5) = ADDR_LR) then
       Func_Strb       <= '1';
     elsif (Cmd_Shift_Reg(8 downto 5) = "1111") then
       Func_Strb       <= '1';
     else
       Func_Strb       <= '0';
     end if;

   else
     GCFE_ADDR       <= GCFE_ADDR;
     Chip_FUNC       <= Chip_FUNC;
     Func_Strb       <= '0';
   end if;

end if;
End Process; -- CMD_SM

DataSTRB: Process(DAQ_CLK,RESET)
Begin
```

```
if (RESET = '1') then
     Data_Strb <= '0';
        Data_Reg  <= Data_Reg;
elsif (DAQ_CLK'event and DAQ_CLK = '1') then

   if ((Ld_DAT_reg = '1') and (GCFE_ADDR = ADDR_LR)) then
     Data_Strb <= '1';
        Data_Reg <= Data_Shift_Reg(15 downto 0);
   else
     Data_Strb <= '0';
        Data_Reg <= Data_Reg;
   end if;
end if;
end process; -- DataStrb

-- SR_Reg Shifts Start and CMD in

SR_Reg: Process(DAQ_CLK, RESET)
Begin
if (RESET = '1') then
    Cmd_Shift_Reg   <= "000000000";
    Data_Shift_Reg  <= "0000000000000000";
elsif (DAQ_CLK'event and DAQ_CLK = '1') then
   Cmd_Shift_Reg(0)    <= CMD;
   Cmd_Shift_reg(8 downto 1) <= Cmd_Shift_Reg(7 downto 0);
    Data_Shift_Reg(0)   <= CMD;
    Data_Shift_Reg(15 downto 1)   <= Data_Shift_Reg(14 downto 0);

end if;

End Process; -- SR_Reg


--
-- Decode the incoming command stream
--
-- Uses the Start Bit shifting though the SR
-- to increament the State Machine
--

-- SROUT <= Shift_Reg(16);

Decode: Process(Next_State, Present_State, CMD, count)
Begin

Case Present_State is

When Idle =>
   Ld_CMD_Reg      <= '0';
   LD_DAT_Reg      <= '0';
   Ld_Count        <= '1';
   Dn_Count        <= '0';
   Next_Count      <= "01001"; -- 9 = 5 command bits 4 address bits
   if (CMD = '1') then
     Next_State  <= CMD_In;
   else
     Next_State  <= Idle;
   end if;
```

```
When CMD_In =>
   LD_DAT_Reg     <= '0';
   if (Count = "00000") then
      Ld_CMD_Reg  <= '1';
      Ld_Count  <= '1';
      Dn_Count  <= '0';
      Next_Count  <= "01111"; -- 15 = 16 data bits
      Next_State  <= DAT_In;
   else
      Ld_CMD_Reg  <= '0';
      Ld_Count  <= '0';
      Dn_Count  <= '1';
      Next_Count  <= "00000";
      Next_State  <= CMD_In;
   end if;

When DAT_In =>
   Ld_CMD_Reg     <= '0';
   Next_Count       <= "00000";

   If (Count = "00000") then
      LD_DAT_Reg  <= '1';
      Ld_Count  <= '1';
      Dn_Count  <= '0';
      Next_State  <= Term;
   else
      Ld_CMD_Reg  <= '0';
         LD_DAT_Reg  <= '0';
      Ld_Count  <= '0';
      Dn_Count  <= '1';
      Next_State  <= DAT_In;
   end if;

When Term =>
   Ld_CMD_Reg     <= '0';

   LD_DAT_Reg     <= '0';
   Ld_Count     <= '0';
   Dn_Count     <= '0';
   Next_Count       <= "00000";
   Next_State     <= Idle;

When others =>
   Ld_CMD_Reg     <= '0';
   LD_DAT_Reg     <= '0';
   Ld_Count     <= '0';
   Dn_Count     <= '0';
   Next_Count       <= "00000";
   Next_State     <= Idle;

end Case;

End Process; -- Decode

end Behaviour; -- CMD_Seq
```

5.5 READ_BACK.VHD

```
--
--
-- Created by David Nelson Feb 8, 2001
--
--  Filename: read_back.vhd
--
--  Function:  Read all of the GCFE Registers listed in the GCFE ASIC specification
--              listed in Section 7, Tables 2, 3, & 4. This code reads the registers
from
--              "GCFE_REG" and sends the data serially to "DIG_MUX" for transport to
the
--              "DIG_DAT_OUT" pins
--
--  Modifications:
--
--
--Library gcfe_vhdl;
Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
--use gcfe_vhdl.gcfe_pkg.all;
use gcfe_pkg.all;

Entity Read_back is
Port   (
  DAQ_CLK     : in std_logic;
  RESET       : in std_logic;
  Func_Strb   : in std_logic;
  Chip_Func   : in std_logic_vector(4 downto 0);
    CTRL_REG_0   : in std_logic_vector(15 downto 0);
    CTRL_REG_1      : in std_logic_vector(15 downto 0);
    FLE_DAC_REG     : in std_logic_vector(15 downto 0);
    FHE_DAC_REG     : in std_logic_vector(15 downto 0);
    LOG_ACPT_DAC_REG : in std_logic_vector(15 downto 0);
    RNG_ULD_DAC_REG  : in std_logic_vector(15 downto 0);
    REF_DAC_REG      : in std_logic_vector(15 downto 0);
  CTRL_DATA     : out std_logic;
    CTRL_SEL        : out std_logic
  );
End Read_back;

Architecture Behaviour of Read_back is

Signal Count     : std_logic_vector(4 downto 0);
Signal Ld_Count    : std_logic;
Signal Dn_Count    : std_logic;

Signal OutSr     : std_logic_Vector(16 downto 0);
Signal Next_OutSr  : std_logic_Vector(15 downto 0);
Signal Ld_OutSr    : std_logic;
Signal En_OutSr    : std_logic;
Signal Ld_Next_Outsr  : std_logic;

constant Reg_Size   : std_logic_vector( 4 downto 0) := "10001";

Type Ctrl_State is
  (
```

```
   Idle,
       Reg_Load,
   Reg_Out,
   Terminate
   );

Signal Present_State, Next_State  : Ctrl_State;

Begin

RD_SM: Process(DAQ_CLK,RESET)
 Begin
   if (RESET = '1') then
      Present_State <= Idle;
   elsif  (DAQ_CLK'event and DAQ_CLK = '1') then
   Present_State <= Next_State;
   end if;
 End Process; -- RD_SM

Cntr: Process(DAQ_CLK,RESET)
 Begin
   if (RESET = '1') then
    Count <= "00000";
   elsif (DAQ_CLK'event and DAQ_CLK = '1') then
   if (Ld_Count = '1') then
     Count <= Reg_Size;
   elsif (Dn_Count = '1') then
     Count <= Count - 1;
   else
     Count <= Count;
   end if;
   end if;
 End Process;  -- Cntr

OutSr_Reg: Process(DAQ_CLK, RESET)
  Begin
    if (RESET = '1') then
       OutSr <= ( others => '0');
       CTRL_DATA        <= '0';

    elsif (DAQ_CLK'event and DAQ_CLK = '1') then

   if (En_OutSR = '1') then
     OutSR(0)            <= '0';
            CTRL_DATA           <=  OutSR(16);
     OutSR(16 downto 1)  <= OutSR(15 downto 0);
   elsif (Ld_OutSr = '1') then
     OutSR(15 downto 0)  <= Next_Outsr;
            OutSR(16)           <= '1';
            CTRL_DATA           <= '0';
   else   OutSR(16 downto 0)  <= OutSR(16 downto 0);
            CTRL_DATA           <= '0';
   end if;
   end if;
  End Process; -- OutSr_Reg

Load_SR: Process(Ld_Next_Outsr, Daq_clk, RESET,
    Chip_Func, CTRL_REG_0, CTRL_REG_1, FLE_DAC_REG, FHE_DAC_REG,
      LOG_ACPT_DAC_REG, RNG_ULD_DAC_REG, REF_DAC_REG )
```

```
  Begin
    if (RESET = '1') then
         Next_Outsr <= "0000000000000000";
    elsif (DAQ_CLK'event and DAQ_CLK = '1') then

      if (Ld_Next_Outsr = '1') then
          case Chip_Func is
           when RD_CONFIG_REG_0_CMD =>
                Next_Outsr <= CTRL_REG_0;
           when RD_CONFIG_REG_1_CMD =>
              Next_Outsr <= CTRL_REG_1;
           when RD_FLE_DAC_CMD =>
                Next_Outsr <= FLE_DAC_REG;
           when RD_FHE_DAC_CMD =>
              Next_Outsr <= FHE_DAC_REG;
           when RD_LOG_ACPT_DAC_CMD =>
                Next_Outsr <= LOG_ACPT_DAC_REG;
           when RD_RNG_ULD_DAC_CMD =>
                Next_Outsr <= RNG_ULD_DAC_REG;
           when RD_REF_DAC_CMD =>
                Next_Outsr <= REF_DAC_REG;
           when others =>
                Next_Outsr <= "0000000000000000";
           end case;
       else
         Next_Outsr  <= Next_Outsr;
      end if;
    end if;
  end process;

Readit: Process(present_state, next_state, Func_Strb, Count,
   Chip_Func)
begin

Case Present_State is
   When Idle =>
  En_OutSr    <= '0';
  Ld_OutSr    <= '0';
      CTRL_SEL    <= '0';
      Dn_Count   <= '0';
      Ld_Count   <= '0';
      Ld_Next_Outsr <= '1';
      if (( Func_Strb = '1' ) AND (Chip_Func(4) = '1' ) )then
    Next_State  <= Reg_Load;
  else
    Next_State  <= Idle;
      end if;

  when Reg_Load =>
    Ld_Count    <= '1';
    En_OutSr    <= '0';
    Ld_OutSr    <= '1';
    CTRL_SEL    <= '1';
    Dn_Count    <= '0';    Ld_Next_Outsr <= '0';

    Next_State  <= Reg_Out;

  When Reg_Out =>
    Ld_OutSr    <= '0';
```

```
    Ld_Count    <= '0';
    En_OutSr    <= '1';
    CTRL_SEL    <= '1';
    Dn_Count    <= '1';
    Ld_Next_Outsr <= '0';

       if (Count = "00000") then
     Next_State  <= Terminate;
   else
     Next_State  <= Reg_Out;
   end if;

  When Terminate =>
   Dn_Count  <= '0';
   Ld_Count  <= '0';
   Ld_OutSr  <= '0';
   En_OutSr  <= '0';
    CTRL_SEL    <= '0';
    Ld_Next_Outsr <= '0';

     Next_State  <= Idle;

  When Others =>
    CTRL_SEL    <= '0';
   Dn_Count  <= '0';
   Ld_Count  <= '0';
   Ld_OutSr  <= '0';
   En_OutSr  <= '0';
    Ld_Next_Outsr <= '0';

   Next_State  <= Idle;

  End Case;

End Process; -- Readit

End Behaviour;
```

5.6 CTRL_ACQ.VHD

```
--
-- Created by David Nelson Jan 24, 2001
--
--
--  Function: Component definition file for GCFE
--
--  This code's function is described in Conceptional Design of the Glast
--  Calorimeter Front-End Electronics (GCFE) ASIC, document #89.
--  section titled Signal Acquisition Control.
--
--  Filename: Ctrl_Acq.vhd
--
--  Function:
--  RESET must be false to start an acquisition
--  RESET will abort ACQ always
--  All actions are on clock true edge
--
--  Wait for START_ACQ
--  Set HOLD if START_ACQ
--  Latch Data from RNG_ULD[2:0]
--  INC_RNG
--  Wait 2 DAQ_CLK
--  INC_RNG
--  Wait 2 DAQ_CLK
--  INC_RNG
--  Wait 2 DAQ_CLK
--  RESET HOLD
--
--  then Shift Din and get the command and adress from input.
--
--  Wait until cmdseq process completes, then wait for start bit again.
--
--  Modifications:
--
--

--Library gcfe_vhdl;
Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use gcfe_pkg.all;


Entity CTRL_ACQ is
Port   (

-- Input signals from GCRC

  DAQ_CLK     : in std_logic;
  START_ACQ       : in std_logic;
    RESET           : in std_logic;

-- Output signals to RANGE_CTRL

    INC_RNG         : out std_logic;
    LD_RNG_ULD      : out std_logic;
    HOLD            : buffer std_logic;
```

```
-- Output signals to DIG_MUX

    SEL_LOG_ACPT    : out std_logic;
    RNG_SEL_1       : out std_logic;
    RNG_SEL_0       : out std_logic
    );
end CTRL_ACQ;

Architecture Behaviour of CTRL_ACQ is

Type ACQ_State is
   (
   State_Idle,
   State_Hold,
    State_LD_ULD,
    State_Log_Acpt,
    State_Rng_1,
    State_Rng_0,
    State_Inc_Rng
   );

Signal Present_State, Next_State  : ACQ_State;

begin

-- CMD_SM Sequences through the Command State Machine

ACQ_SM: process(RESET, DAQ_CLK)
    Begin
    if(RESET = '1') then
        Present_State <= State_Idle;
    elsif (DAQ_CLK'event and DAQ_CLK = '1') then
      Present_State <= Next_State;
    end if;
End Process; -- ACQ_SM

Decode: Process(Next_State, Present_State, START_ACQ)
Begin

Case Present_State is

When State_Idle =>
   HOLD         <= '0';
   LD_RNG_ULD   <= '0';
    SEL_LOG_ACPT <= '0';
    RNG_SEL_1    <= '0';
    RNG_SEL_0    <= '0';
    INC_RNG      <= '0';
    if (START_ACQ = '1') then
        Next_State  <= State_Hold;
    else
        Next_State  <= State_Idle;
    end if;

when State_Hold =>
    HOLD         <= '1';
    LD_RNG_ULD   <= '0';
```

```
    RNG_SEL_1    <= '0';
    RNG_SEL_0    <= '0';
    INC_RNG      <= '0';
    SEL_LOG_ACPT <= '0';
    Next_State   <= State_LD_ULD;

when State_LD_ULD =>
    HOLD         <= '1';
    SEL_LOG_ACPT <= '0';
    RNG_SEL_1    <= '0';
    RNG_SEL_0    <= '0';
    INC_RNG      <= '0';
    LD_RNG_ULD   <= '1';
    Next_State   <=  State_Log_Acpt;

when State_Log_Acpt =>
    HOLD            <= '1';
    LD_RNG_ULD      <= '0';
    SEL_LOG_ACPT    <= '1';
    RNG_SEL_1       <= '0';
    RNG_SEL_0       <= '0';
    INC_RNG         <= '0';
    Next_State      <= State_Rng_1;

when State_Rng_1 =>  -- we loop back here from State_Inc_Rng for 4 ranges

    HOLD         <= '1';
    LD_RNG_ULD   <= '0';
    SEL_LOG_ACPT <= '0';
    RNG_SEL_1    <= '1';
    RNG_SEL_0    <= '0';
    INC_RNG      <= '0';
    Next_State   <= State_Rng_0;

when State_Rng_0 =>
    HOLD            <= '1';
    SEL_LOG_ACPT    <= '0';
    LD_RNG_ULD      <= '0';
    INC_RNG         <= '0';
    RNG_SEL_1       <= '0';
    RNG_SEL_0       <= '1';
    Next_State      <= State_Inc_Rng;

when State_Inc_Rng =>
    HOLD            <= '1';
    SEL_LOG_ACPT    <= '0';
    LD_RNG_ULD      <= '0';
    RNG_SEL_1       <= '0';
    RNG_SEL_0       <= '0';
    INC_RNG         <= '1';
    Next_State      <= State_Rng_1;

when others =>
   HOLD         <= '0';
   LD_RNG_ULD   <= '0';
    SEL_LOG_ACPT <= '0';
    RNG_SEL_1    <= '0';
    RNG_SEL_0    <= '0';
    INC_RNG      <= '0';
```

```
    Next_State   <= State_Idle;

end case;
end process; -- decode
end Behaviour; -- CTRL_ACQ;
```

5.7 RANGE_CTRL.VHD

```
--
-- Created by David Nelson Jan 24, 2001
--
--  Filename: range_cntrl.vhd
--
--  Function: Multiplexor
--  Send DIG_IN out to DIG_DAT_OUT if CTRL_SEL is true
--  Send LOG_ACPT_LLD to DIG_DAT_OUT if CTRL_SEL is false and
--  RNG_SEL(0)
--
--  Send RANGE(0) to DIG_DAT_OUT if CTRL_SEL is false and RNG_SEL(1)
--  Send RANGE(1) to DIG_DAT_OUT if CTRL_SEL is false and RNG_SEL(2)
--  RANGE(3) is undefined.
--
--  Modifications:
--
--

--Library gcfe_vhdl;
Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use gcfe_pkg.all;


Entity Range_cntrl is
Port   (

-- Input signals from CTRL_CONFIG

    RESET          : in  std_logic;
    DAQ_CLK         : in  std_logic;
    USE_FRST_RNG    : in  std_logic;
   OVERWRITE       : in  std_logic;
    FIRST_RNG       : in  std_logic_vector(1 downto 0);
    ENA_RNG         : in  std_logic_vector(1 downto 0);

-- Input signals from CTRL_ACQ

    INC_RNG         : in std_logic;
    LD_RNG_ULD      : in std_logic;

-- Input signals from CTRL_ACQ

-- Input signal from RANGE_MUX

    RNG_ULD         : in std_logic_vector(2 downto 0);

-- Output signal to RANGE_MUX

    RANGE_SEL       : buffer std_logic_vector(4 downto 0);

-- Output to DIG_MUX

    RANGE_VOLT      : out std_logic_vector(1 downto 0)

   );
```

```
End Range_cntrl;

Architecture Behavior of Range_cntrl is

    constant RNG_ULD_0 : std_logic_vector(3 downto 0) := "0001";
    constant RNG_ULD_1 : std_logic_vector(3 downto 0) := "0010";
    constant RNG_ULD_2 : std_logic_vector(3 downto 0) := "0100";
    constant RNG_ULD_3 : std_logic_vector(3 downto 0) := "1000";


    constant FIRST_RNG_LEX8 : std_logic_vector(1 downto 0) := "00";
    constant FIRST_RNG_LEX1 : std_logic_vector(1 downto 0) := "01";
    constant FIRST_RNG_HEX8 : std_logic_vector(1 downto 0) := "10";
    constant FIRST_RNG_HEX1 : std_logic_vector(1 downto 0) := "11";

    constant Select_Ref  : std_logic_vector(4 downto 0) := "10000";
    constant Select_HEX1 : std_logic_vector(4 downto 0) := "01000";
    constant Select_HEX8 : std_logic_vector(4 downto 0) := "00100";
    constant Select_LEX1 : std_logic_vector(4 downto 0) := "00010";
    constant Select_LEX8 : std_logic_vector(4 downto 0) := "00001";


    signal First_out :    std_logic_vector(4 downto 0) ;

    signal RNG_SEL_RST  : std_logic;

    signal Lat_ULD_0    : std_logic;
    signal Lat_ULD_1    : std_logic;
    signal Lat_ULD_2    : std_logic;

    signal S_First_range    : std_logic;
    signal S_Second_range    : std_logic;
    signal S_Third_range    : std_logic;
    signal S_Fourth_range    : std_logic;

Type Range_State is
  (
  Idle,
  SM_First_Range,
    SM_Second_Range,
    SM_Third_Range,
    SM_Fourth_Range
  );

Signal Present_State, Next_State  : Range_State;

Begin

RNG_SEL_RST <= ( RESET );

-- Range_SM Sequences through the Range State Machine

Range_SM: Process(RESET, DAQ_CLK)
    Begin
     if (RESET = '1') then
        Present_State <= Idle;
     elsif (DAQ_CLK'event and DAQ_CLK = '1') then
     Present_State <= Next_State;
```

```
    end if;
    End Process; -- Range_SM

Set_Range: Process(Next_State, Present_State, INC_RNG, LD_RNG_ULD)
Begin

Case Present_State is

When Idle =>
       S_First_Range  <= '0';
       S_Second_Range <= '0';
       S_Third_Range  <= '0';
       S_Fourth_Range <= '0';

       if (Ld_RNG_ULD = '1') then
          Next_state <= SM_First_range;
       else
          Next_state <= Idle;
       end if;

When SM_First_Range =>
       S_First_Range  <= '1';
       S_Second_Range <= '0';
       S_Third_Range  <= '0';
       S_Fourth_Range <= '0';

       if (Inc_Rng = '1') then
          Next_state <= SM_Second_range;
       else
          Next_state <= SM_First_range;
       end if;

When SM_Second_Range =>
       S_First_Range  <= '0';
       S_Second_Range <= '1';
       S_Third_Range  <= '0';
       S_Fourth_Range <= '0';

       if (Inc_Rng = '1') then
          Next_state <= SM_Third_range;
       else
          Next_state <= SM_Second_range;
       end if;


When SM_Third_Range =>
       S_First_Range  <= '0';
       S_Second_Range <= '0';
       S_Third_Range  <= '1';
       S_Fourth_Range <= '0';

       if (Inc_Rng = '1') then
          Next_state <= SM_Fourth_range;
       else
          Next_state <= SM_Third_range;
       end if;

When SM_Fourth_Range =>
       S_First_Range  <= '0';
```

```
       S_Second_Range <= '0';
       S_Third_Range  <= '0';
       S_Fourth_Range <= '1';

       if (Inc_Rng = '1') then
          Next_state <= Idle;
       else
          Next_state <= SM_Fourth_range;
       end if;

When Others =>
       S_First_Range  <= '0';
       S_Second_Range <= '0';
       S_Third_Range  <= '0';
       S_Fourth_Range <= '0';

       Next_state <= Idle;

end case;

end Process;  -- Set_Range



 -- Next determines which is the first range to output when ENA_RNG is not 0
Range_determine: process (ENA_RNG,Lat_ULD_0, Lat_ULD_1, Lat_ULD_2 )
Begin

  case ENA_RNG is
                        when "00"   =>
                            First_Out  <= Select_Ref;
                        when "01"   =>
                            if ( Lat_ULD_0 = '1') then
                                First_Out <= Select_LEX1;
                            else
                                First_Out <= Select_LEX8;
                            end if;
                        when "10"   =>
                            if (Lat_ULD_2 = '1') then
                               First_Out <= Select_HEX1;
                            else
                                First_Out <= Select_HEX8;
                            end if;
                        when "11"   =>
                            if (Lat_ULD_2 = '1') then
                                First_Out <= Select_HEX1;
                            elsif (Lat_ULD_1 = '1') then
                                First_Out <= Select_HEX8;
                            elsif (Lat_ULD_0 = '1') then
                                First_Out <= Select_LEX1;
                            else
                                First_Out <= Select_LEX8;
                            end if;
                        when others =>
                            First_Out <= Select_Ref;
  end case; -- ena_rng

end process;
```

```
Mux_range: Process(USE_FRST_RNG, OVERWRITE, FIRST_RNG, Rng_sel_rst,
                     First_out, S_First_range, S_Second_range,
                      S_Third_range, S_Fourth_Range)

Begin

if (OVERWRITE = '1') then
     if (USE_FRST_RNG = '0') then
                RANGE_SEL <= Select_Ref;
     else
        case FIRST_RNG is
                    when "00" =>
                        RANGE_SEL <= Select_LEX8;
                    when "01" =>
                        RANGE_SEL <= Select_LEX1;
                    when "10" =>
                        RANGE_SEL <= Select_HEX8;
                    when "11" =>
                        RANGE_SEL <= Select_HEX1;
                    when others =>
                        RANGE_SEL <= Select_Ref;
         end case;
      end if;
 else   -- OVERWRITE = 0
        if (RNG_SEL_RST = '1') then
                RANGE_SEL <= Select_Ref;
        else -- OVERWRITE = 0 and RNG_SEL_RST = 0


           if (S_First_range = '1') then
                 if (USE_FRST_RNG = '1') then
                    case FIRST_RNG is
                    when "00" =>
                        RANGE_SEL  <= Select_LEX8;
                    when "01" =>
                        RANGE_SEL  <= Select_LEX1;
                    when "10" =>
                        RANGE_SEL  <= Select_HEX8;
                    when "11" =>
                        RANGE_SEL  <= Select_HEX1;
                    when others =>
                        RANGE_SEL  <= Select_Ref;
                    end case;
                 else -- OVERWRITE = 0 and RNG_SEL_RST = 0 and USE_FRST_RNG = 0
                    if (First_out = Select_LEX8 ) then
                            RANGE_SEL  <= Select_LEX8;
                    elsif (First_out = Select_LEX1 ) then
                            RANGE_SEL  <= Select_LEX1;
                    elsif (First_out = Select_HEX8 ) then
                            RANGE_SEL  <= Select_HEX8;
                    else
                            RANGE_SEL  <= Select_HEX1;
                    end if;
                 end if; -- use_frst_range =1

           elsif ( S_Second_range = '1') then
                 if (USE_FRST_RNG = '1') then
                    case FIRST_RNG is
```

```
                when "00" =>
                    RANGE_SEL  <= Select_LEX1;
                when "01" =>
                    RANGE_SEL  <= Select_HEX8;
                when "10" =>
                    RANGE_SEL  <= Select_HEX1;
                when "11" =>
                    RANGE_SEL  <= Select_LEX8;
                when others =>
                    RANGE_SEL  <= Select_Ref;
                end case;
            else -- OVERWRITE = 0 and RNG_SEL_RST = 0 and USE_FRST_RNG = 0
                if (First_out = Select_LEX8 ) then
                        RANGE_SEL  <= Select_LEX1;
                elsif (First_out = Select_LEX1 ) then
                        RANGE_SEL  <= Select_HEX8;
                elsif (First_out = Select_HEX8 ) then
                        RANGE_SEL  <= Select_HEX1;
                else
                        RANGE_SEL  <= Select_LEX8;
                end if;
            end if; -- use_frst_range =1

        elsif ( S_Third_range = '1')  then
            if (USE_FRST_RNG = '1') then
                case FIRST_RNG is
                when "00" =>
                    RANGE_SEL  <= Select_HEX8;
                when "01" =>
                    RANGE_SEL  <= Select_HEX1;
                when "10" =>
                    RANGE_SEL  <= Select_LEX8;
                when "11" =>
                    RANGE_SEL  <= Select_LEX1;
                when others =>
                    RANGE_SEL  <= Select_Ref;
                end case;
            else -- OVERWRITE = 0 and RNG_SEL_RST = 0 and USE_FRST_RNG = 0
                if (First_out = Select_LEX8 ) then
                        RANGE_SEL  <= Select_HEX8;
                elsif (First_out = Select_LEX1 ) then
                        RANGE_SEL  <= Select_HEX1;
                elsif (First_out = Select_HEX8 ) then
                        RANGE_SEL  <= Select_LEX8;
                else
                        RANGE_SEL  <= Select_LEX1;
                end if;
            end if; -- use_frst_range =1

        elsif ( S_Fourth_range = '1') then
            if (USE_FRST_RNG = '1') then
                case FIRST_RNG is
                when "00" =>
                    RANGE_SEL  <= Select_HEX1;
                when "01" =>
                    RANGE_SEL  <= Select_LEX8;
                when "10" =>
                    RANGE_SEL  <= Select_LEX1;
                when "11" =>
```

```
                        RANGE_SEL  <= Select_HEX8;
                    when others =>
                        RANGE_SEL  <= Select_Ref;
                    end case;
                else -- OVERWRITE = 0 and RNG_SEL_RST = 0 and USE_FRST_RNG = 0
                    if (First_out = Select_LEX8 ) then
                            RANGE_SEL  <= Select_HEX1;
                    elsif (First_out = Select_LEX1 ) then
                            RANGE_SEL  <= Select_LEX8;
                    elsif (First_out = Select_HEX8 ) then
                            RANGE_SEL  <= Select_LEX1;
                    else
                            RANGE_SEL  <= Select_HEX8;
                    end if;
                end if; -- use_frst_range =1


        else
                RANGE_SEL <= Select_Ref;
        end if; -- sm_state= first range
         end if; -- rng_sel_rst = 1
   end if; -- overwrite = 1


end process;

Dig_Outputs: process(RANGE_SEL)
begin

   case RANGE_SEL is
                    when Select_LEX8 =>
                        RANGE_VOLT <= "00";
                    when Select_LEX1 =>
                        RANGE_VOLT <= "01";
                    when Select_HEX8 =>
                        RANGE_VOLT <= "10";
                    when Select_HEX1 =>
                        RANGE_VOLT <= "11";
                    when others =>
                       RANGE_VOLT <= "00";
    end case;
 end process;

Lat_ULDs: Process(DAQ_CLK, LD_RNG_ULD,RESET)
Begin
if(RESET = '1') then
        Lat_ULD_0 <= '0';
        Lat_ULD_1 <= '0';
        Lat_ULD_2 <= '0';
elsif (DAQ_CLK'event and DAQ_CLK = '1') then
    if (LD_RNG_ULD = '1') then
        Lat_ULD_0 <= RNG_ULD(0);
        Lat_ULD_1 <= RNG_ULD(1);
        Lat_ULD_2 <= RNG_ULD(2);
    else
        Lat_ULD_0 <= Lat_ULD_0;
        Lat_ULD_1 <= Lat_ULD_1;
        Lat_ULD_2 <= Lat_ULD_2;
    end if;
```

```
end if;

end process; -- Lat_ULDs

end Behavior; -- Range_cntrl
```

5.8 DIG_MUX.VHDL

```
--
-- Created by David Nelson Jan 24, 2001
--
--  Filename: dig_mux.vhd
--
--  Function: Multiplexor
--  Send DIG_IN out to DIG_DAT_OUT if CTRL_SEL is true
--
--  Send RANGE(0) to DIG_DAT_OUT if CTRL_SEL is false and  HOLD is
--  false and RNG_SEL(1)
--  Send RANGE(1) to DIG_DAT_OUT if CTRL_SEL is false and RNG_SEL(2)
--  RANGE(3) is undefined.
--
--  Modifications:
--
--
--Library gcfe_vhdl;
Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use gcfe_pkg.all;


Entity Dig_Mux is
Port   (

-- Input signals from CTRL_CONFIG

    CTRL_SEL        : in std_logic;
   CTRL_DATA       : in std_logic;
    SEL_LOG_ACPT    : in std_logic;

-- Input signals from CTRL_ACQ

    RNG_SEL_1       : in std_logic;
    RNG_SEL_0       : in std_logic;

-- Input signals from RANGE_CNTRL

    RANGE_VOLT      : in std_logic_vector( 1 downto 0 );

-- Input signal from RANGE_MUX

    LOG_ACPT_LLD    : in std_logic;

-- Output to GCRC

    DIG_DAT_OUT     : buffer std_logic

   );

End Dig_Mux;

Architecture Behavior of Dig_Mux is

Begin
Data_Mux: process(CTRL_SEL, RNG_SEL_1, RNG_SEL_0, CTRL_DATA,
```

```
                    SEL_LOG_ACPT, LOG_ACPT_LLD, RANGE_VOLT)
    begin
        if (CTRL_SEL = '1') then
            DIG_DAT_OUT <= CTRL_DATA;
        else
            if (SEL_LOG_ACPT = '1') then
                DIG_DAT_OUT <= LOG_ACPT_LLD;
            elsif (RNG_SEL_1 = '1') then
                DIG_DAT_OUT <= RANGE_VOLT(1);
            elsif
                (RNG_SEL_0 = '1') then
                DIG_DAT_OUT <= RANGE_VOLT(0);
            else
                DIG_DAT_OUT <= '0';
            end if;
        end if;
    end process; -- Data_Mux
end Behavior; -- Dig_Mux
```