KEITHLEY

Application Note Series

Number 2457

Making Charge-Pumping Measurements with the Model 4200-SCS Semiconductor Characterization System

Overview of the Charge-Pumping Technique

Charge-pumping measurements are widely used to characterize interface state densities in MOSFET devices. Recently, with the development of high dielectric (high κ) gate materials, charge-pumping has proven especially useful in characterizing charge-trapping phenomena in high κ thin gate films. In thin gate films, leakage current is relatively high due to quantum mechanical tunneling of carriers through the gate. As a result, the traditional technique for extracting interface trap density—collecting simultaneous quasistatic and high frequency C-V measurement data and comparing the difference—can't be used because quasistatic C-V is very hard to achieve at the leakage current level. However, charge-pumping measurements can still be used to extract interface trap density, and the effect of gate leakage can be compensated for by measuring charge-pumping current at lower frequency and subtracting it from measurement results at higher frequencies [1, 2].

The basic charge-pumping technique involves measuring the substrate current while applying voltage pulses of fixed amplitude, rise time, fall time, and frequency to the gate of the transistor, with the source, drain, and body tied to ground. The pulse can be applied with a fixed amplitude, voltage base sweep or a fixed base, variable amplitude sweep.

In a voltage base sweep, the amplitude and period (width) of the pulse are fixed while sweeping the pulse base voltage (***Figure 1a***). At each base voltage, body current can be measured and plotted against base voltage. The interface trap density (D_{it}) can be extracted as a function of bandbending, based on this equation:

$$D_{it} = \frac{I_{cp}}{qAf\Delta E}$$

where I_{cp} is the measured charge-pumping current, q is the fundamental electronic charge, A is the area, f is the frequency, and ΔE is the difference between the inversion Fermi level and the accumulation Fermi level [3].

A fixed base, variable amplitude sweep has a fixed base voltage and pulse frequency with step changes in voltage amplitude (***Figure 1b***). The information obtained is similar to that extracted from a voltage base sweep. These measurements can also be performed at different frequencies to obtain a frequency response for the interface traps.

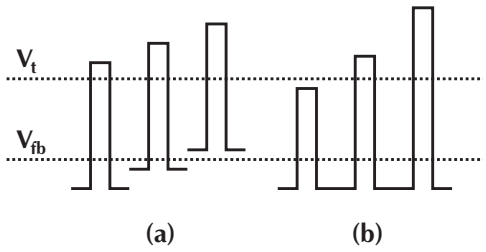


Figure 1. Overview of charge-pumping measurements:

(a) Pulse waveform for base voltage sweep; pulse amplitude is constant.

(b) Pulse waveform for amplitude sweep; base voltage is constant.

Hardware setup

It's relatively easy to perform charge-pumping measurements and data analysis using a Model 4200-SCS (Semiconductor Characterization System) in combination with a pulse generator (such as Agilent's Model 8110, 81110, or 8112). The KTE Interactive software that runs the Model 4200-SCS can simultaneously control the system's internal Source-Measure Units (SMUs) and external instruments via GPIB with simple C programming. Refer to the Model 4200-SCS Reference Manual and Keithley Application Notes for guidance on using the Model 4200-SCS and KTE Interactive software. ***Figure 2a*** illustrates the connections for a device under test (DUT) with one of the Model 4200-SCS's SMUs and a pulse generator without a switch matrix; in ***Figure 2b***, a semiconductor switch matrix is included in the configuration. This application note describes how to perform charge-pumping measurements with the Model 4200-SCS and an Agilent Model 8110 pulse generator.

Installing the Charge-Pumping Measurement Driver

A driver for making charge-pumping measurements with the Model 4200-SCS can be obtained from Keithley Instruments. Currently, this driver supports Agilent Model 8112 and Series 8110/81110 pulse generators. Be sure to request the driver written specifically for the pulse generator model to be used. The driver contains modules that can do base-sweep and amplitude-sweep. To install the driver on 4200-SCS, copy the source files (.c files) to a new temp folder. Then, go to the **Windows menu bar**, select **Run** and type *cmd*. In the DOS prompt window that pops up, change the directory to the temp folder that contains the

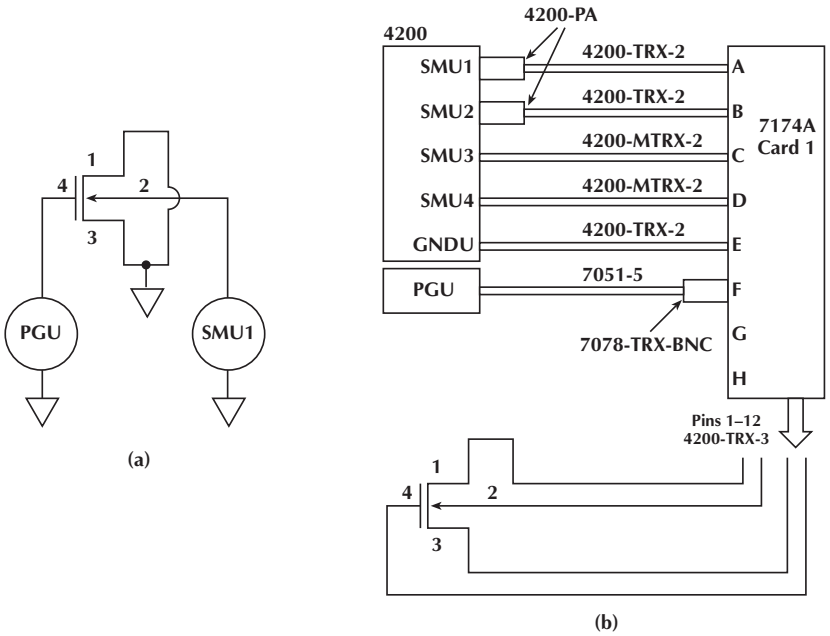


Figure 2. DUT connection with 4200-SCS and pulse generator
(a) DUT connection without a switch matrix
(b) DUT connection with switch matrix

source files, and type in *kultcopy ChargePumping* and *enter*. This will install the driver on the Model 4200-SCS.

Note: The driver is currently unsupported. Please exercise caution in using it.

Setting Up KCON (Keithley CONfiguration utility)

The first thing to do is to enter the proper pulse generator model number (and switch matrix, if necessary) into KCON, the software interface that controls the Model 4200-SCS's internal hardware (SMUs and preamps) and external instruments with GPIB communication. To access KCON, just double-click on **KCON** on the Model 4200-SCS desktop. Then, from the **Tools** menu, select **Add external Instruments > Pulse Generator > HP8110/81110**. ***Figure 3*** shows a KCON window with this pulse generator added. If the pulse generator model to be used is not included on the supported list, it must be added as a "General Purpose Instrument" (GPI) rather than a "Pulse Generator" (PGU) in KCON. After the pulse generator is added, KCON assigns an instrument ID string to the pulse generator. This ID string could be *PGUx* or *GPIx*, depending on how the pulse generator is added (Pulse Generator Unit or General Purpose Instrument), and *x* could be any number from 1 to 4. This ID will be used as an input parameter for the charge-pumping measurement. It tells KITE software which instrument is on the GPIB and its address.

Setting Up a Project in KITE

KITE is the main software interface that controls internal hardware and external instruments with GPIB interfaces. Charge-pumping measurement will be performed in KITE interface with a UTM (User Test Module). Refer to the Model 4200-SCS Reference Manual for more details on KITE operation and UTMs.

Double-click on the **KITE** icon on the Model 4200-SCS desktop to bring up the KITE interface. From the **menu bar**, select **File > New project** and type in the project name. Then, go once more to the **menu bar** and select **Project > Make new subsite plan** and type in a valid subsite name. Next, go to the **menu bar** and select **Project > Make new device plan** and choose a device from the MOSFET folder. Finally, go to the **menu bar** and select **Project > Make new User Test Module** and type in the module name, (for example, *BaseSweep*) and

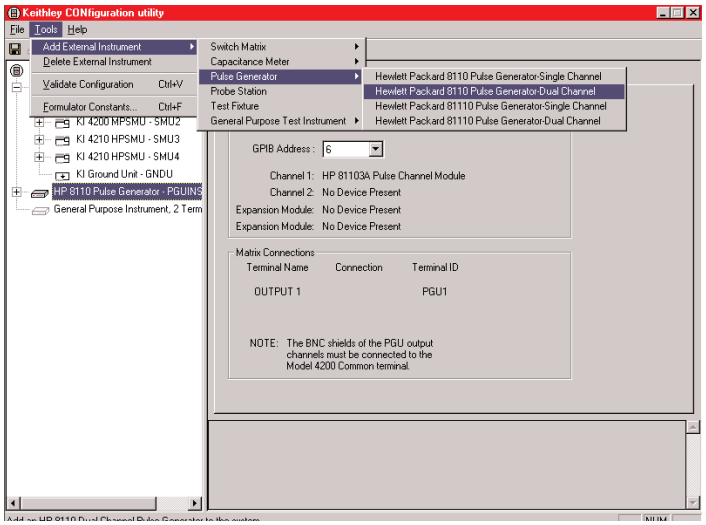


Figure 3. KCON setup window

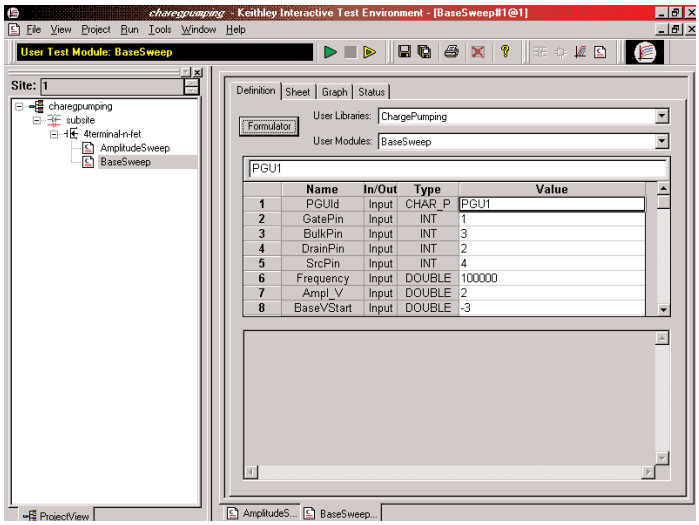
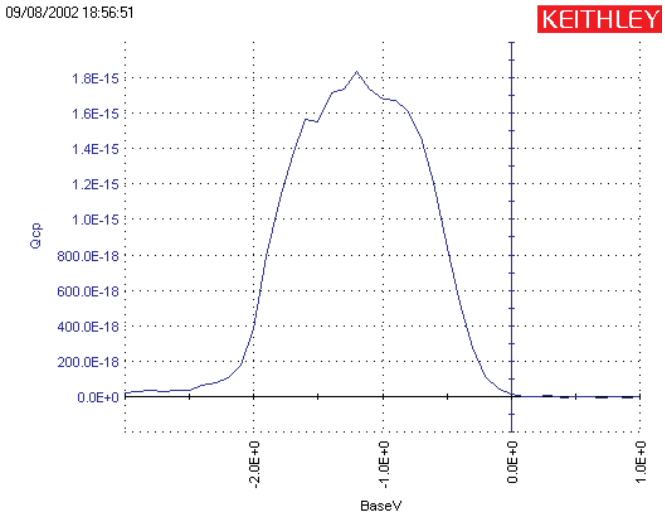
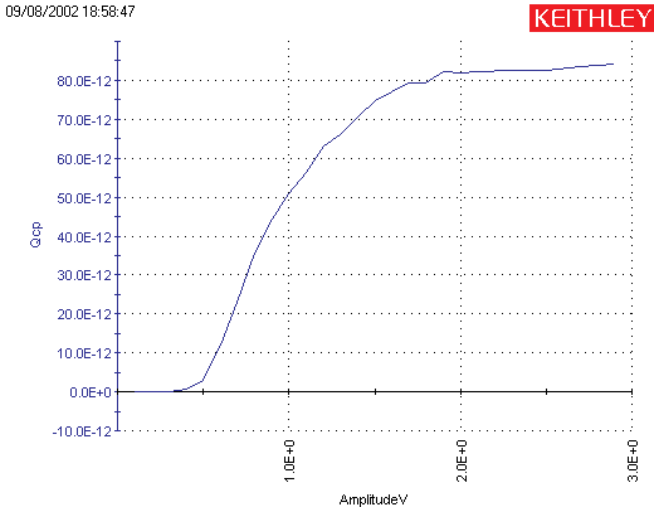


Figure 4. KITE project window

click **OK**. Double-click the **BaseSweep** module in the project tree and a setup window will appear. In the setup window, choose user library **ChargePumping** from the drop-down library list, and choose module **BaseSweep** from the User Module list. After the module is selected, a parameter window will appear, as in ***Figure 4***. The first parameter on the list is *PGUId*, which refers to the ID in KCON. If HP8110 is used and properly configured in KCON, *PGU1* should be used. If HP8112 is used, then GPIx should be used, where x is a number assigned in KCON. Fill in the parameter list with the proper setup parameters, such as switch connection (if no switch is presented, then fill in *0* in the *GatePin*, *BulkPin*, *SourcePin* and *DrainPin* field), frequency, pulse amplitude, and duty cycle. After all the input parameters have been properly defined, click the **Save** button (the small floppy disk icon) and the test is ready to run. Click the **Run** button (Green triangle icon) to execute the test. The data can be plotted in graph form once the measurement is complete. ***Figure 5*** illustrates two examples of these graphs.



(a)



(b)

Figure 5. Example plot. (a) Base voltage sweep (b) Amplitude sweep

Calculating D_{IT}

The Model 4200-SCS's Formulator function supports calculating D_{IT}. To activate the Formulator window, click the **Formulator** button on the definition tab of a test setup window. Enter the formula for D_{IT}, as shown in ***Figure 6***. The resulting D_{IT} value can be plotted with the system's graphing tools.

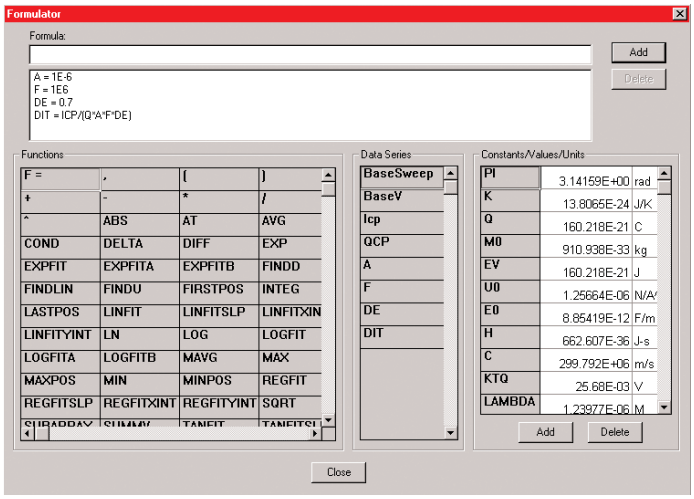


Figure 6. Entering formulas in the Formulator.

Conclusion

The KTE Interactive software on the Model 4200-SCS makes it very easy to make charge-pumping measurements with a pulse generator. With the current driver, one does not need any programming work to do charge-pumping with an Agilent 8112 or 8110/81110 pulse generator. Simple data analysis can be done in the built-in Formulator and plotted with the powerful graphing tools. This makes the 4200-SCS an ideal tool for characterizing interface properties of gate dielectrics, especially in the area of high dielectric material development.

References

[1] P. Masson, et al., "On the Tunneling Component of Charge Pumping Current in Ultrathin Gate Oxide MOSFETs," *IEEE Elect. Dev. Lett.*, Vol. 20, No. 2, pp. 92-94, 1999.

[2] Chung, Steve S., et al., "A Novel and Direct Determination of the Interface Traps in Sub-100nm CMOS Devices with Direct Tunneling Regime (12~16Å) Gate Oxide," *2002 VLSI Tech. Digest of Tech. Papers*.

[3] G. Groeseneken, H.E. Maes, N. Beltran, and R.F. De Keersmaecker, "A Reliable Approach to Charge-Pumping Measurements in MOS Transistors," *IEEE Trans. Electron. Dev.*, Vol. ED-31, pp. 42-53, 1984.

Appendix 1:
Example source code for base sweep using 8110

```
#include "keithley.h"

int BaseSweep( char *PGUId, int GatePin, int BulkPin, int
DrainPin, int SrcPin, double Frequency, double Ampl_V, double
BaseVStart, double BaseVStop, double BaseVStep, double
RiseTime, double FallTime, double DutyCycle, double LoadImp,
double *BaseV, int BaseVSize, double *Icp, int IcpSize, double
*Qcp, int QcpSize )
{
/* USRLIB MODULE CODE */
int index;
int NumPoints;
int PGU1;
char CommandString[50];
int fcnstat;
char  TempBuf[20] = "";
int GPIBAddress;
char GPIBAddressStr[10];
double idummy;
int temp = 1;
int PLC;

getinstid(PGUId, &PGU1);                                    //
get PGU id from KCON

if (PGU1 < 0)
    return( INVAL_INST_ID );                        // No such
PGU

getinstattr(PGU1, "GPIBADDR", GPIBAddressStr);
GPIBAddress = atoi( GPIBAddressStr );

if (PguInit(GPIBAddress)< 0) return -1;                  //
Initialize PGU
// set up PGU
if (PguSetup(GPIBAddress, RiseTime, FallTime,DutyCycle,
Frequency, LoadImp)<0) return -1;

//Validate Input parameters
if (GatePin > 72)  return(INVAL_PARAM);                   //
Validate pins
if (BulkPin > 72)  return(INVAL_PARAM);                   //
Validate pins
if (SrcPin > 72)  return(INVAL_PARAM);                    //
Validate pins
if (DrainPin > 72)  return(INVAL_PARAM);                  //
Validate pins
if (Frequency == 0)  return(INVAL_PARAM);
// Validate pins
if (BaseVStep == 0)  return(INVAL_PARAM);
// Validate pins

//Initialize return arrays
for (index = 0; index < IcpSize; index ++)
{
    BaseV[index] = DBL_NAN;
    Icp[index] = DBL_NAN;
    Qcp[index] = DBL_NAN;
}

// setup switch matrix if necessary
if(GatePin > 0)
    conpin(PGU1,GatePin, 0);
if(BulkPin > 0)
    conpin(SMU1,BulkPin, 0);
if(SrcPin > 0)
    conpin(SMU2,SrcPin, 0);
if(DrainPin > 0)
    conpin(SMU2,DrainPin, 0);

//Setup SMU
forcev(SMU2, 0);
forcev(SMU1,0);
lorangei(SMU1,1e-10);
if (Frequency >= 1e6)
    PLC = 1;
else if (Frequency >= 1e5)
    PLC = 2;
else if (Frequency >= 1e4)
    PLC = 5;
else
    PLC= 10;

//Initialize current range
setmode(SMU1,KI_INTGPLC,PLC);
limiti(SMU1,1e-2);
measi(SMU1, &idummy);

NumPoints =(int) fabs((BaseVStart - BaseVStop) / BaseVStep) +
1;

// Turn on output on Pulse Generator
sprintf(CommandString, ":OUTPUT1 ON\n");
fcnstat = kibsnd(GPIBAddress, -1, GPIBTIMO,
strlen(CommandString), CommandString );
if (fcnstat > 0) return(GPIB_ERROR_OCCURED);

//Main sweep loop
for (index = 0; index < NumPoints; index++)
{
    BaseV[index] = BaseVStart + index * BaseVStep;
    // Program the pulse height. This one is tricky...so get
it right!
    if ( Ampl_V > 0)
    {
        sprintf( CommandString, ":VOLT1:LOW
%9.3e\n",BaseV[index]);
        fcnstat = kibsnd(GPIBAddress, -1, GPIBTIMO,
strlen(CommandString), CommandString );
        sprintf( CommandString, ":VOLT1:HIGH %9.3eV\n",
Ampl_V+BaseV[index]);
        fcnstat = kibsnd(GPIBAddress, -1, GPIBTIMO,
strlen(CommandString), CommandString );

    }
    else {
        sprintf( CommandString, ":VOLT1:LOW %9.3eV\n",
Ampl_V+BaseV[index]);
        fcnstat = kibsnd(GPIBAddress, -1, GPIBTIMO,
strlen(CommandString), CommandString );
        sprintf( CommandString, ":VOLT1:HIGH
%9.3eV\n",BaseV[index]);
        fcnstat = kibsnd(GPIBAddress, -1, GPIBTIMO,
strlen(CommandString), CommandString );
        sprintf( CommandString, ":OUTP1:POL INV\n");
        fcnstat = kibsnd(GPIBAddress, -1, GPIBTIMO,
strlen(CommandString), CommandString );
    }
    if (fcnstat > 0) return(GPIB_ERROR_OCCURED);

    // Measure substrate current
    if (index == 0)  delay(200);
    measi(SMU1, &idummy);
    delay(10);
    intgi(SMU1, &Icp[index]);
    Qcp[index] = Icp[index]/Frequency;
}
//turn off output
sprintf(CommandString, ":OUTPUT1 OFF\n");
fcnstat = kibsnd(GPIBAddress, -1, GPIBTIMO,
strlen(CommandString), CommandString );
if (fcnstat > 0) return(GPIB_ERROR_OCCURED);

return OK;
/* USRLIB MODULE END  */
}                                                                /*
End BaseSweep.c */
```


KEITHLEY

Keithley Instruments, Inc. 28775 Aurora Road • Cleveland, Ohio 44139 • 440-248-0400 • Fax: 440-248-6168
1-888-KEITHLEY (534-8453) • www.keithley.com

No. 2457
8032kGW