MUXDAC Data Source User Guide

UG6904; Rev 0; 4/19

Abstract

This document is an installation and usage guide of the MUXDAC Data Source (MDS). The MDS is a collection of software, firmware, and hardware that provide the digital stimulus to Maxim's RF DACs that employ parallel, multiplexed LVDS data interfaces. The tool supports the MAX19692/MAX19693, MAX5879, and MAX5882 evaluation kits.

Table of Contents

List of Figures

List of Tables

General Description

The MUXDAC Data Source (MDS) is an FPGA-based digital pattern generator tool developed for the evaluation of Maxim's RF DAC products that employ a multiplexed LVDS digital interface. The tool has the following three primary components:

- The MUXDAC EV Kit Software Controller (MDS GUI), which is a Microsoft® Windows® 7 or Windows 10-compatible application and provides a simple graphical user interface (GUI).
- Custom FPGA configuration firmware.
- A user supplied Xilinx® Virtex®-7 FPGA VC707 evaluation kit.

Communication between the PC and the MicroBlaze™ core of the FPGA on the VC707 is achieved through a USB 2.0 interface, which provides fast pattern downloads and full operational control of the system.

The MDS provides four 14-bit LVDS ports to the FMC1 HPC connector on the VC707. Three additional signal types are also supported and provide the following functionality:

- A loop-back, differential clock to compensate for downstream delays.
- A differential data clock signal for a frequency reference by the FPGA.
- Single-ended CMOS signals for EV kit specific controls.

The system operates in load synchronous mode to utilize the data clock signal provided by the DAC EV kit. Supported word rates range from 125Mwps up to 1250Mwps on each of the four multiplexed digital ports.

The 1GB on-board memory of the VC707 and the design of the memory management system can store multiple test patterns simultaneously. The selected pattern is output on the FMC1 HPC connector in a continuous looping mode. The memory controller ensures that the data stream is never interrupted, which is critical for glitch-free operation of the DAC under test. The simple GUI interface provides all the necessary controls to load patterns and select the active pattern from the list and the playback start and stop functions. Additional controls for configuring the FPGA and timing adjustments on the digital interface are also provided.

Features and Benefits

- Proven Software and Hardware Solution
- Commercial, Off-the-Shelf, FPGA Evaluation Board
- Rapid System Development and Prototyping
- Windows 7 and Windows 10 Compatible GUI Software
- Up to 1250Mwps Interface Speed
- 4:1 or 2:1 Multiplexed Interfaces
- High-Speed Pattern Transfer through a USB 2.0 Interface
- Supports the following RF DAC Evaluation Kits (EV kits):
    - MAX19692 EV kit
    - MAX19693 EV kit
    - MAX5879 EV kit
    - MAX5882 EV kit

Required Equipment

- Windows PC (Windows 7 or Windows 10 recommended) with two available USB 2.0 interfaces
- Compatible Maxim Integrated™ RF DAC EV kit
- Xilinx Virtex-7 FPGA VC707 EV kit
    - VC707 board
    - 12V/5A power cube
    - One USB-A to Mini-B cable for interfacing and programming
    - One USB-A to Micro-B cable for interfacing and programming
- DAC clock source
    - RF signal generator or similar (user supplied)
    - Clock source must be able to drive a 50Ω load
- RF signal observation
    - Spectrum analyzer or other user-supplied methods
    - AC coupled, 50Ω source impedance

Required Software Components

- MUXDACEVKITSoftwareController.exe (download from **www.maximintegrated.com/**)
- Xilinx ISE 14.7 Lab Tools (download from **www.xilinx.com/**)
- .NET Framework 4 (included in the MUXDACEVKITSoftwareController.exe installation)
- libusb0 Driver (included in the MUXDACEVKITSoftwareController.exe installation)

Software Installation

Installation of the MDS software is a multi-step process. Detailed instructions for download and installation of all software components are provided in the ***MDS Installation Instructions*** section.

Quick Start

The start-up of the MDS involves the following steps:

- Hardware Setup – Refer to the RF DAC EV kit data sheet for detailed requirements
- Launch the MDS GUI application
- Configure the FPGA
- Select the RF DAC EV kit in use
- Load test patterns
- Start the pattern
- Observe the DAC performance

The quick start procedures are included in the EV kit data sheets.

System Overview

The MDS hardware provides a platform for system design and performance verification under targeted operating conditions.

Figure 1 shows the system block diagram.

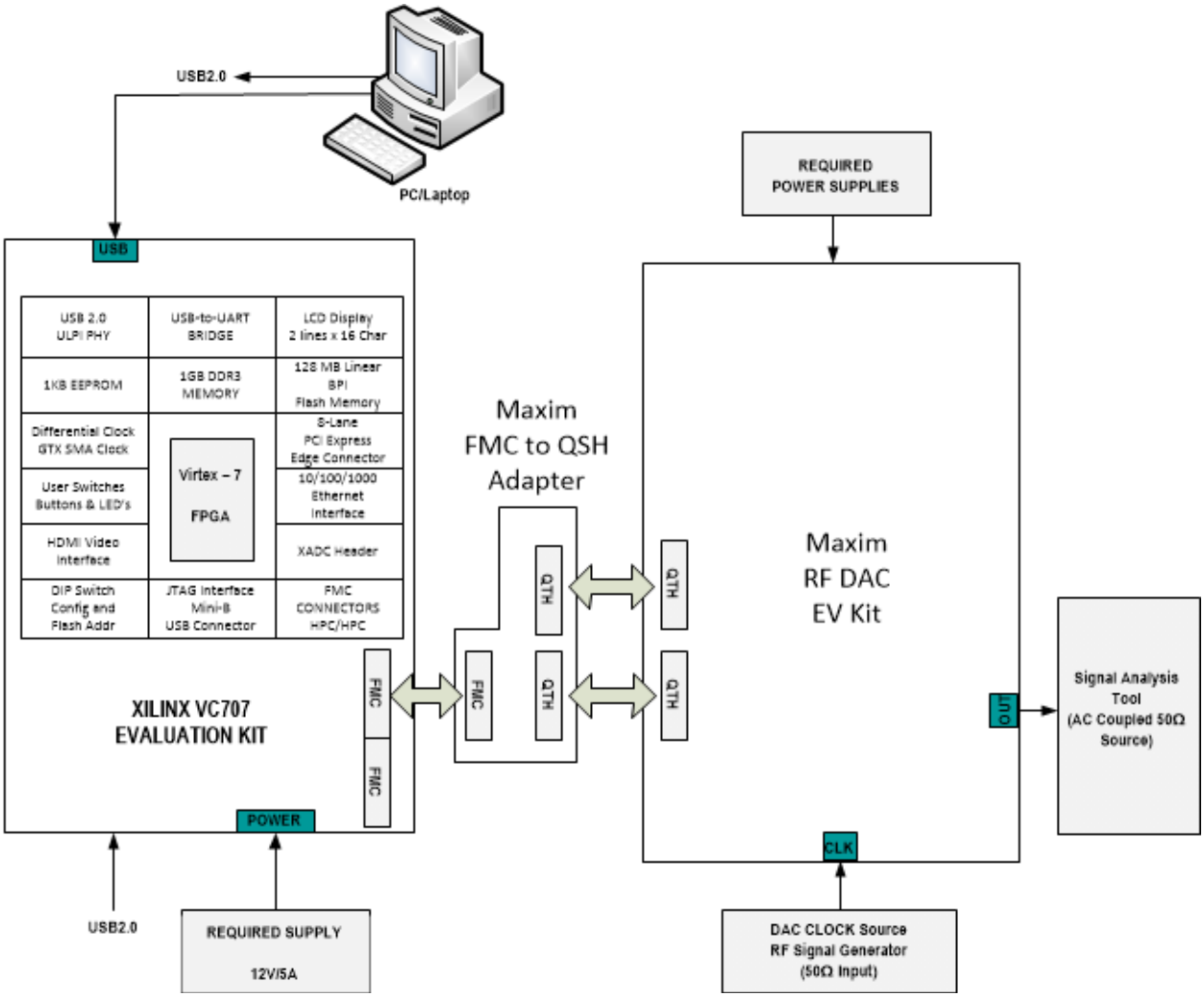


Figure 1. MDS system.

The digital interface of the MDS system employs the FMC1 HPC connector on the VC707 and has the following features:

- 56 pairs of LVDS data signals arranged in four ports of 14-bits each.
- A single LVDS signal pair for the data clock (DATACLK or DCLK) signal from the EV kit in use.
- Two pairs of LVDS signals for loopback timing measurement on the supported EV kits.
- 3.3V CMOS signals for controlling the EV kit.

The 4:1 multiplexed interface supports DAC update rates from 500Msps to 5000Msps. The 2:1 mode is rated from 500Msps to 2500Msps.

The FPGA firmware (Core) enables the USB 2.0 interface on the VC707. Operational commands and pattern data are transferred between the PC and the FPGA through this interface. The Core employs Xilinx's MicroBlaze soft processor core, which processes commands to manage the system memory for pattern storage and playback as well as the operational state of the data interface.

The Core memory manager stores the test patterns in contiguous memory to help minimize memory read access times which is critical for high-speed playback. The test pattern is output to the digital interface in continuous playback mode, and the memory read process is optimized to prevent missing data. The 1GB memory of the VC707 allows for simultaneous storage of multiple test patterns. See the ***Pattern Lists and Pattern Files*** section for details.

The MDS GUI configures and communicates with the VC707 to provide operational control of the hardware. The FPGA firmware can be loaded with each power-up of the VC707 by using the ISE 14.7 Lab Tools and the **Load FPGA Configuration File** button in the GUI. Optionally, the firmware can be automatically loaded at power-up by using the **Load EEPROM Image** button. Connecting the USB 2.0 interface enables loading test patterns, adjusting interface timing, and controlling the playback.

Detailed Software Description

The MDS GUI controls the MDS operation. The software starts with a splash screen, as shown in **Figure 2**.

Figure 2. MDS GUI splash screen.

Select the checkbox to disable the splash screen on future start-ups of the MDS GUI.

The main GUI window appears after initialization completes, as shown in **Figure 3**.

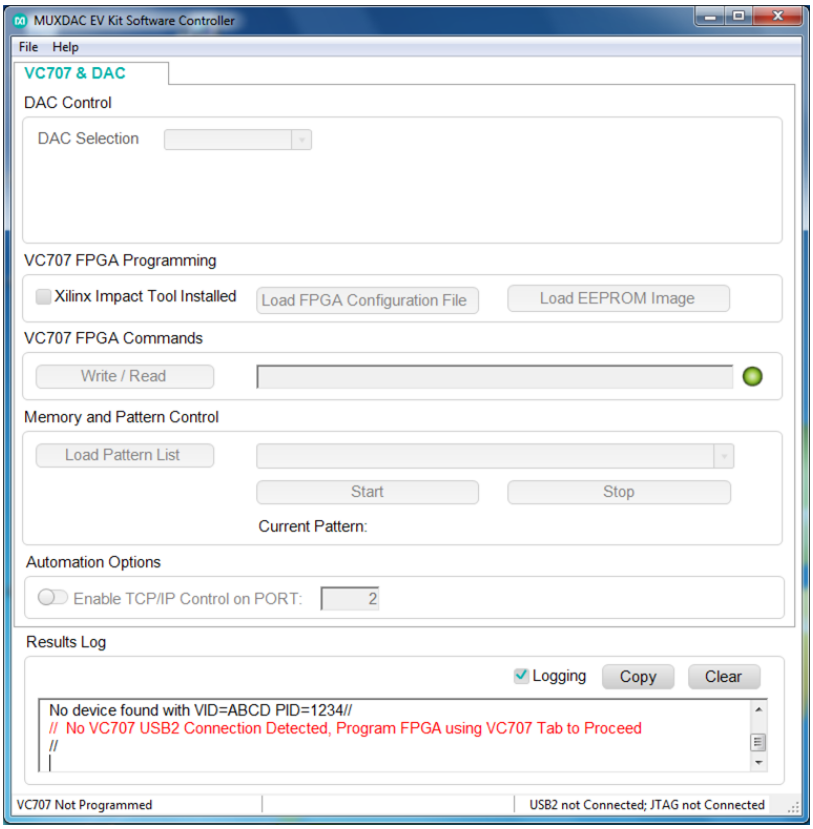


Figure 3. Application window.

The controls are separated into sections within the window. The sections from top to bottom are as follows:

- DAC Control
- VC707 FPGA Programming
- VC707 FPGA Commands
- Memory and Pattern Control
- Automation Options
- Results Log

Additionally, the lower edge of the window provides the live status of the FPGA programming (lower left) as well as the USB 2.0 and JTAG connections (lower right).

DAC Control

In the DAC Control section, select the EV kit in use from the drop-down list of the supported EV kits, as shown in **Figure 4**.

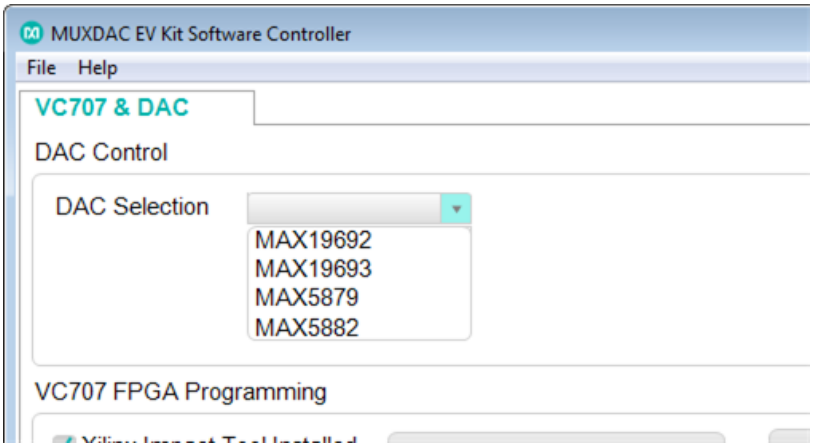
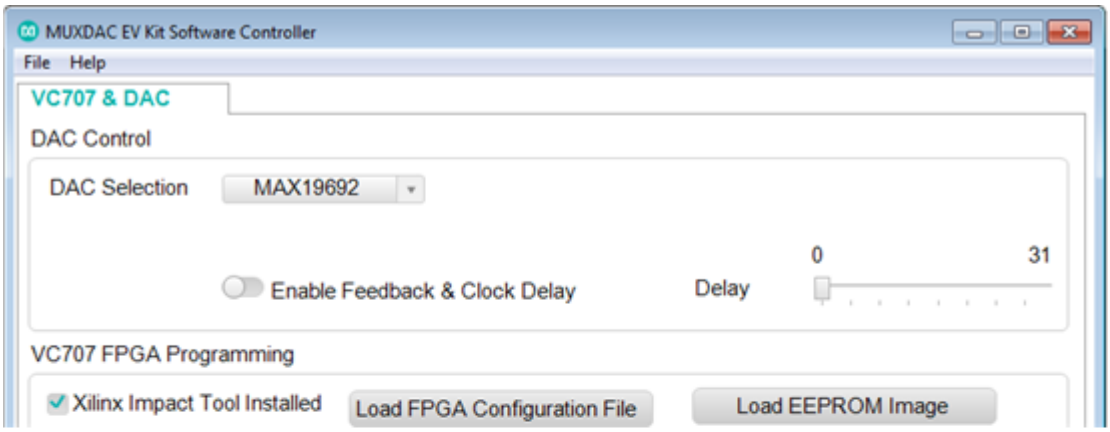


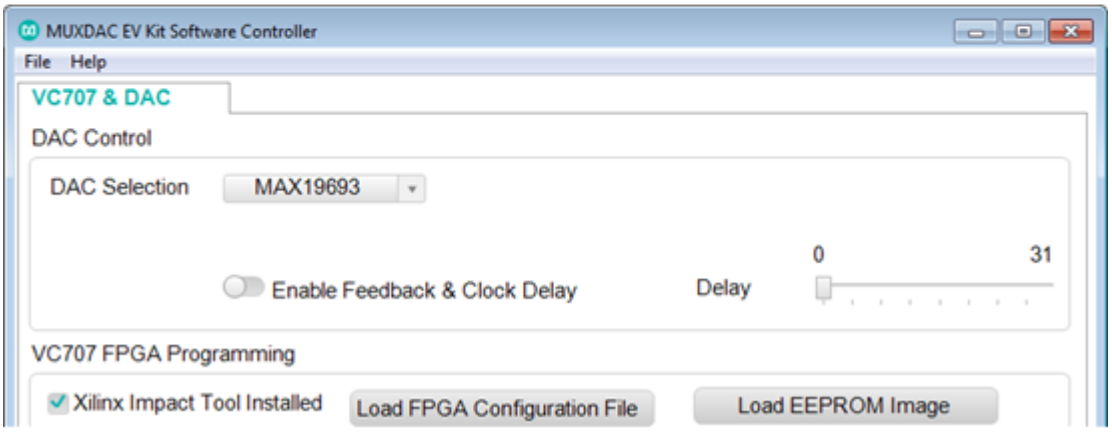
Figure 4. DAC selection.

Additional controls are populated within the DAC Control section depending on the EV kit selected. The MAX19692 and MAX19693 are 12-bit devices with setup and hold time requirements for the input data relative to their DATACLK output signal. With the MDS, the user can adjust the relative timing of the FPGA data output relative to the DCLK input signal to optimize the interface timing and achieve the best possible DAC performance.

Figure 5 shows the DAC Control section when the MAX19692 (a) or MAX19693 (b) is selected. See the ***MUXDAC FPGA Command Set*** section for details of the commands used for DAC timing control.



a) MAX19692



b) MAX19693

Figure 5. 12-bit DAC timing controls for the MAX19692 and MAX19693.

The MAX5879 and MAX5882 are 14-bit devices that include a delay locked loop (DLL), which eases the timing adjustments to ensure optimal performance. The user can disable the DLL and manually adjust the timing on these devices. The MAX5879 also supports two separate 2:1 MUX modes.

Figure 6 and **Figure 7** show the various options for the MAX5879 and MAX5882, respectively.

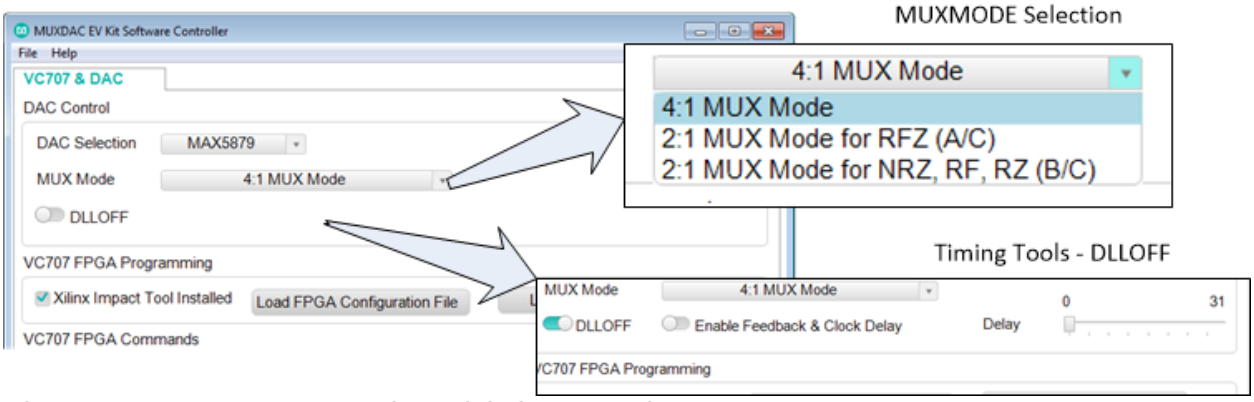


Figure 6. MAX5879 MUX mode and timing controls.

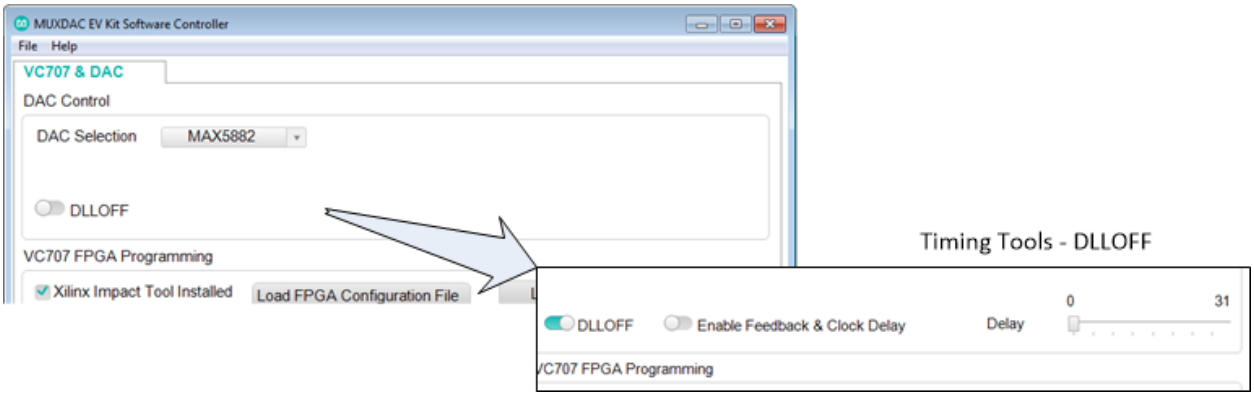


Figure 7. MAX5882 timing controls.

VC707 FPGA Programming

Program the FPGA by using the controls provided in the VC707 FPGA Programming section of the GUI.

At start-up, the **Xilinx Impact Tool Installed** checkbox is the only active control in this window. Two buttons are also visible: **Load FPGA Configuration File** and **Load EEPROM Image**. These buttons are not activated until the JTAG communication is confirmed and the **Xilinx Impact Tool Installed** box is checked. The controls are shown in Figure 3.

When this control is first activated (by clicking the checkbox), the program requests the user to locate the Impact tool, which performs the actual FPGA configuration. After locating the Impact tool, the system checks the JTAG status. If the JTAG status is connected, the system activates the two buttons.

The **Load FPGA Configuration File** option uses a .bit file that is streamed directly to the FPGA. The use of .bit file programming is volatile and must be performed each time MDS is started. The use of this method is recommended if the VC707 EV kit is used for other projects or developments.

Alternatively, the FPGA image can be stored in EEPROM on the VC707 by clicking the **Load EEPROM Image** button, which transfers a .mcs file to the EEPROM for use at VC707 power-up. The primary benefit of this method is a reduction in start-up time because the VC707 is programmed at power-up. When the GUI establishes communication, the user can proceed by selecting the DAC and loading patterns.

VC707 FPGA Commands

The VC707 FPGA Commands section of the GUI provides the ability to manually communicate with the VC707. When the USB 2.0 is connected, the Commands section is enabled for reporting the response from the last command sent to the FPGA. Simply click in the text box, type the desired command, and click the **Write/Read** button. The command sent to the FPGA and the response are recorded in the Results Log at the bottom of the window. See the ***MUXDAC FPGA Command Set*** section regarding the use of manual commands.

Memory and Pattern Control

Loading test patterns, selecting the desired output pattern, and starting and then stopping the pattern are all accomplished using the controls in the Memory and Pattern Control section. When programming and DAC selection are complete, patterns can be loaded into the memory.

The MDS system uses text files to provide a list of pattern files to load. The MDS GUI installation includes several sample patterns and pattern lists for simple sine wave test patterns. These patterns are arranged into four lists: two for 12-bit devices such as the MAX19692 and MAX19693 and two for 14-bit devices like the MAX5879 and MAX5882.

Figure 8 shows the File Explorer window that opens after clicking the **Load Pattern List** button.

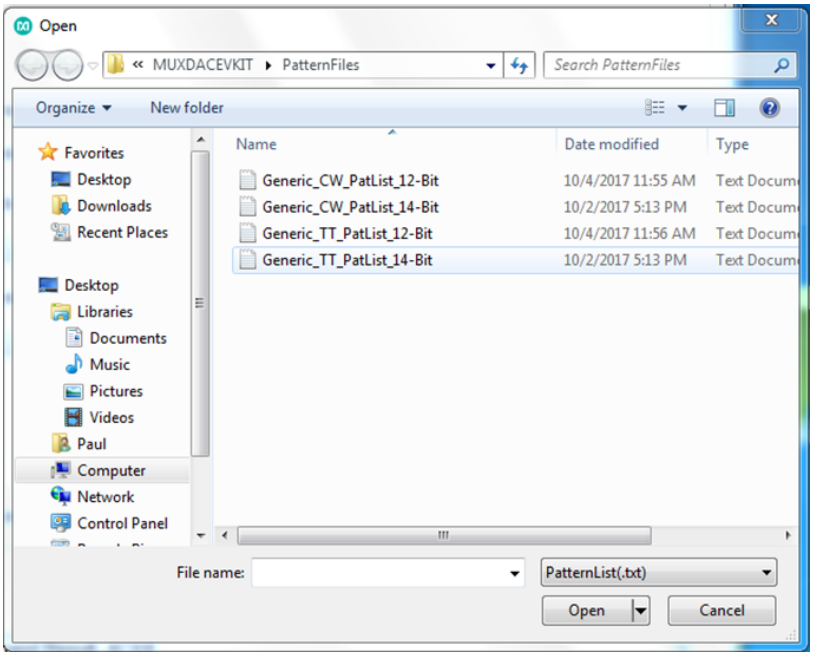


Figure 8. Sample MUXDAC pattern lists.

Select and open one of the two pattern lists that match the EV kit. The GUI displays a circular progress bar that indicates the rough percentage of completion based on the number of patterns loading, as shown in **Figure 9**. The lists with more and/or longer test patterns require more time to transfer.

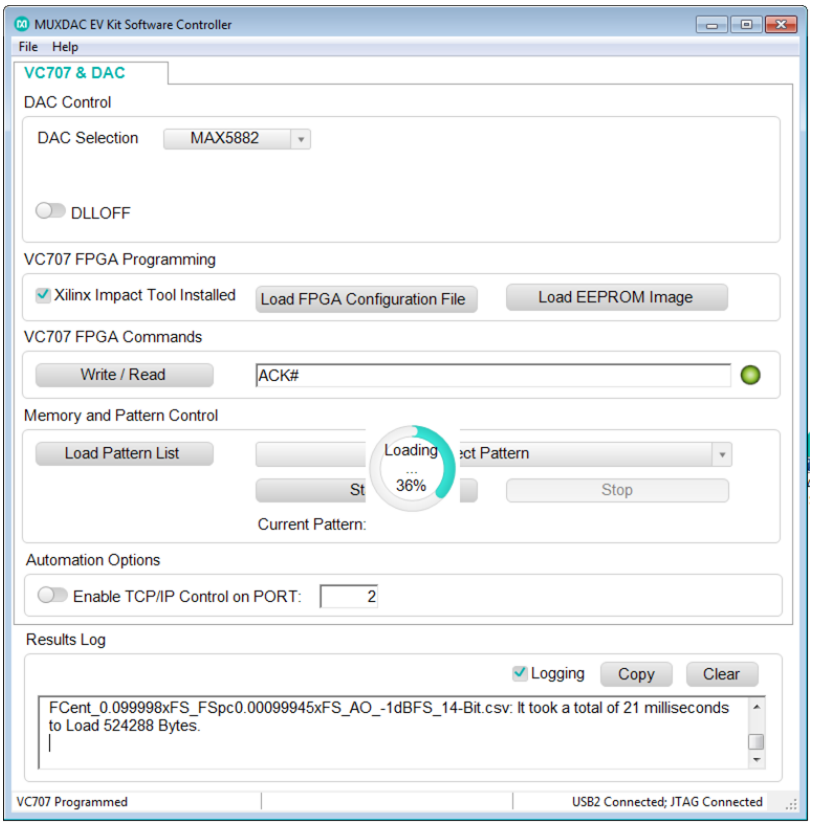


Figure 9. Pattern load in progress.

When the load process is complete, the **Select Pattern** drop-down list populates with pattern file names (minus the extension). To select a pattern, click the drop-down list and then click on the desired pattern.

Figure 10 shows the list loaded when selecting Generic_TT_PatList_14-Bit.txt. The length of this list requires a slider on the right for navigation. Drag the slider up or down to find the desired pattern in the list.

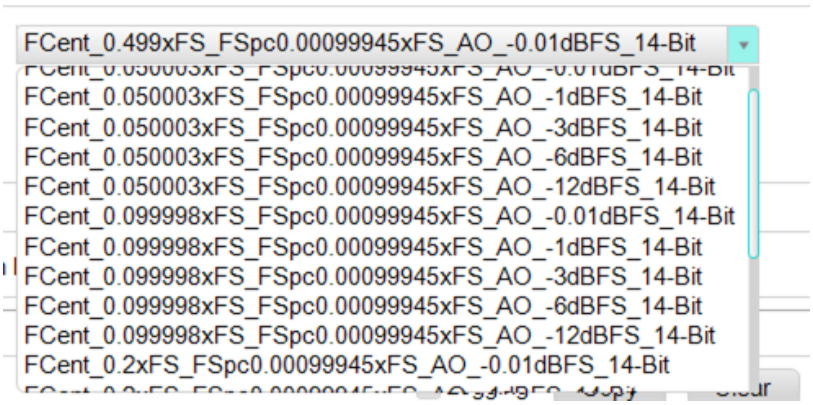


Figure 10. Selecting a test pattern.

After the pattern is selected, click the **Start** button to begin playback. The MDS executes all the necessary initialization steps and then begins to stream the pattern, which continues until the **Stop** button is clicked or the DCLK signal is interrupted.

Figure 11 shows the GUI when playback is active.

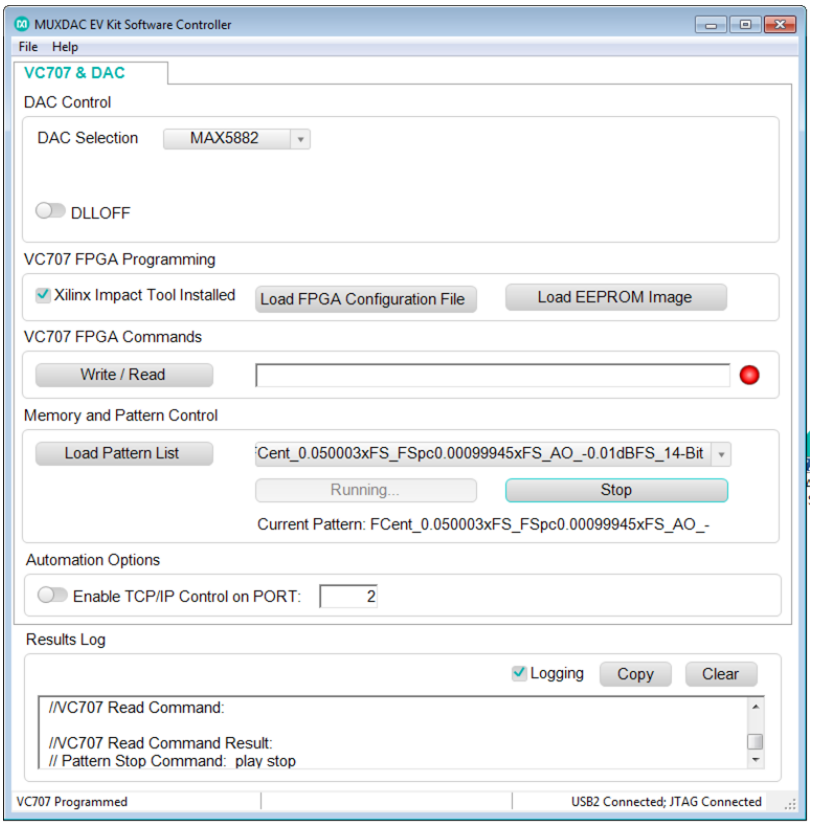


Figure 11. GUI display during playback.

Automation Options

Use the Automation Options section of the GUI to enable external control through a TCP/IP port. The user can automate measurements such as sweeping the DAC output frequency or amplitude by entering a valid TCP/IP port number within the range of 0 to 65535 and then enabling this function.

NOTE: When enabled, the GUI is locked out of local control. Only an RTL command received through the TCP/IP interface returns local control.

See the ***Automating RF DAC Measurements with the MDS GUI*** section for more details.

Results Log

The Results Log displays commands, command responses, and error messages that result from interaction with the hardware. Options are available to disable logging, copying or clearing the window contents. Some information, especially error messages, can still be displayed when the **Logging** option is unchecked.

Pattern Lists and Pattern Files

Pattern List Files

The MDS GUI utilizes list files for loading test patterns into the VC707 memory. Examples of these list files are included in the MUXDACEVKIT\PatternFiles folder, which is automatically created during installation. These files list the names of the test pattern files including the extension. The format of the list file is plain text with one file name on each line. The list can contain multiple patterns with up to 1GB in total pattern length, but only one pattern list can be loaded at a time. Loading a new list causes the previously loaded patterns to be overwritten. The user can omit a file during the load process by adding the '#' character anywhere in the line. The GUI recognizes this character and skips the line when completing the load process.

The MDS application is provided with continuous-wave (CW) and two-tone (TT) test patterns to help the user get the system up and running quickly. Additionally, the patterns are grouped into four list files:

- Generic_CW_PatList_12-Bit.txt
- Generic_TT_PatList_12-Bit.txt
- Generic_CW_PatList_14-Bit.txt
- Generic_TT_PatList_14-Bit.txt.

The sample patterns are also stored in the PatternsFiles folder of the installation directory. There are two pattern name formats: one for single-tone CW signals and one for TT patterns.

An example of the CW pattern name is FO_0.499xFS_AO_-0.01dBFS_12-Bit.csv. The output frequency is 0.499 × the DAC update rate (FS). The amplitude is given in dBFS and is backed off by at least 0.01dB from full scale. The bus widths are also defined in the file name.

The TT patterns use a slightly different naming convention, such as FCent_0.2xFS_FSpc0.00099945xFS_AO_-0.01dBFS_12-Bit.csv. The two tones are centered at FCent, which in this case is 0.2 × FS. In this case, the spacing (FSpc) between the two tones is 0.001 × FS. The amplitude and resolution are handled the same as the CW pattern file names.

Pattern Requirements

Test patterns are a fundamental requirement of the MDS System. These patterns contain the time sequence values that are presented to the DAC under testing. The total number of data points within a given pattern must be an integer multiple of 1024. The actual size of the pattern depends on the user's requirement for the output signal. Many signal analysis tools require a continuous signal for accurate measurements. Therefore, the pattern should transition seamlessly when looping back to the start. Seamless loopback is accomplished when the pattern contains an integer number of cycles of the output frequency. The combination of the seamless loopback requirement and finite pattern lengths typically results in a slight frequency error between the target frequency and actual frequency supported by the pattern length.

Pattern Generation

The MDS GUI comes with several sample patterns for generating sine wave CW and TT signals. Typically, the user generates signals with other properties including modulated signals. User created patterns need to first meet the pattern requirements (see the ***Pattern Requirements*** section), and then be properly formatted for use in the system.

The MDS expects a specific format within the pattern file. The value in the first line is a count of the number of data lines (N) within the file, starting with the second line in the file. The pattern data is arranged in an N × 4 array (N rows, 4 columns), where columns are separated by a comma (.csv format). The pattern sequence starts in Row 1 of the array with Columns 1 and 2 (R1C1/C2) and contains 2 bytes of data representing a 16-bit offset binary value. The data pattern is converted from a single vector of 12- or 14-bit decimal samples into bytes. The least significant byte (LSByte) is stored in columns C1 and C3. The most significant bytes (MSByte) are stored in C2 and C4. The byte values are extracted using an AND operation. The LSByte is the result of the sample value and 0x00FF. The MSByte is derived from the sample value and 0xFF00, and then the result shifts 8 bits to the right.

The pattern continues with R1C3/C4 followed by R2C1/C2 and R2C3/C4. The sequence concludes with RN-1C3/C4, RNC1/C2, and RNC3/C4. **Table 1** shows an example of pattern file contents.

Table 1. Example Pattern File Content

FILE LINE NUMBER (NOT PART OF FILE)	CONTENTS
1 (N*)	65536
2	19, 242, 253, 127
3	19, 242, 250, 127
4	19, 242, 248, 127
5	19, 242, 245, 127
…	…
65534	19, 242, 242, 127
65535	19, 242, 239, 127
65536	19, 242, 236, 127
65537 (N* + 1)	19, 242, 234, 127

(N* is the total number of lines in the data array)

MATLAB Pattern Generation for the MDS

This section includes sample MATLAB® routines that allow a user with MATLAB installed to create additional pattern files as well as continuous wave, sinuous test patterns. Generating test patterns with modulated signals is beyond the scope of this document.

The following MATLAB compatible code is provided without warranty. It is only provided for reference purposes, and the determination of suitability for use is solely the responsibility of the user.

Three software routines are provided:

- Sinewave Pattern Generation
- Pattern Scaling
- Pattern Storage

These routines are targeted at the MDS system and the supported RF DAC EV kits.

Sinewave Pattern Generation

The function for creating sinewave patterns, GenerateSinewave, requires three input parameters:

- The DAC output update rate (f_{DAC}) in Hertz
- The desired output frequency list (f_{TARGET}) in Hertz
- The number of data points in the output pattern (N*)

Note: The pattern length, N, must be an integer multiple of 1024.

The outputs from this function are as follows:

- The pattern data (PatData)
- The calculated output frequency list (f_{OUT}) in Hertz
- The middle point of f_{OUT} (f_{MID}) in Hertz

The PatData output is a vector with length N. The real values of the pattern are scaled between -1.0 and +1.0.

The f_{MID} output is primarily for use with multi-tone patterns, providing the user with the center point of the output frequencies, and is useful for setting the spectrum analyzer. It is calculated from the minimum and maximum frequencies in the f_{OUT} list using the following equation:

$f_{MID} = (\max(f_{OUT}) - \min(f_{OUT}))/2 + \min(f_{OUT})$

The f_{OUT} variable contains the actual output frequencies generated by the pattern. These frequencies are typically slightly off from the user's desired output frequencies. The limitation arises from the need for a continuous, glitch free data stream when playback transitions from the end back to the beginning of the pattern (seamless playback). The minimum output frequency for a given f_{DAC} and pattern length, N, is calculated as the ratio f_{DAC}/N. The frequencies calculated by the function are odd, integer multiples of this minimum frequency.

MATLAB is a registered trademark of The MathWorks Inc.

For example, create a pattern at 4.6Gsps that outputs 500MHz and 550MHz tones and has 32768 data points. The call to the GenerateSinewave routine looks like this:

```
[PatData,fMID,fOUT] = GenerateSinewave(4.6e9, [500e6, 550e6], 32768)
```

The output from this example function call and PatData are as follows:

f_{MID} = 525.164794922MHz

f_{OUT} = 500.177001953MHz, 550.152587891MHz

The frequency errors for the two output tones are 177kHz and 152kHz, and the center has shifted from 525MHz by 164.8kHz, the result of the combination of the minimum frequency and the forced use of an odd frequency multiplier. Even frequency multipliers are not used as they tend to create patterns that only use a small set of repeating DAC input values, regardless of the pattern length. Odd multipliers result in a larger utilization of the available code set, and as the pattern length increases, so does the utilization.

```
function [PatData,fMID,fOUT] = GenerateSinewave(fDAC, fTARGET, N)
% Input variables are:
% fDAC - sample rate in hertz
% fTARGET - target frequency in hertz. Use an array for multiple tones.
% N - number of data points to generate
% Output variables are:
% PatData - vector containing real values in the range -1 to +1
% fMID - center frequency, in Hz, for two tone patterns
% fOUT - output frequency (array for multi-tone patterns)
tonenum=length(fTARGET); % How many tones?
fOUT=fTARGET; % Format fOUT to match fTARGET size
mp=fTARGET; % Format mp to match fTARGET size (one mp per fOUT)

for i=1:tonenum %calculate # of cycles in the pattern
m=floor ((fTARGET(i) / fs) * N +0.5);
if mod(m,2)==0 % check to ensure odd value
mp(i)=m+1; % if even, add 1.
else
mp(i)=m; % if odd, then use this value for tone (i)
end
ft = mp(i)/(N *fDAC); %calculate ft for coherent sampling
fOUT(i)=ft;
```

```
end
if tonenum>1 %if multi-tone, then calculate fMID
fMID=(max(fOUT)-min(fOUT))/2 + min(fOUT);
else
fMID=fOUT(tonenum);
end
PatData(1:numpnts)=0; % initialize the pattern to all zero values
k=1:numpnts; % integer representation of time, allows array math
for i=1:tonenum
PatData=PatData+cos(2.0*pi*k*mp(i)/N); % add tones into one
pattern
end
PatData=PatData/max(abs(PatData)); % rescale values between -1
and 1
end
```

Pattern Scaling

Once we have the basic pattern, it must be converted into binary data for the DAC input. The ScalePattern routine demonstrates the conversion of the real values in the source pattern to 12- or 14-bit integers. The new values are integer representations of the Offset Binary code and range from 0 to 2N - 1, where N is the number of bits. The resolution of the converted pattern should match the EV kit in use.

The input arguments for the scaling routine are as follows:

- The pattern to be scaled (PatIn) within -1.0 to +1.0 limits
- The pattern resolution (N_bits), typically 12 or 14
- The scale relative to DAC full-scale (dBFS), a negative value

Considering the dBFS parameter, 0dBFS is the maximum signal amplitude and refers to an input pattern that utilizes the entire DAC full-scale range. The only option for adjustment of the output signal is to reduce the DAC range utilized, hence the negative value requirement for the dBFS argument.

The only output from this function is the scaled data pattern, ScaledData, ready to be stored in a pattern file. However, the routine displays the minimum and maximum values in the output pattern.

An example of using the script to convert the previously generated pattern, PatData, uses the function call as follows:

```
ScaledData = ScalePattern(PatData,14,-0.001)
```

The output pattern from the above function call is scaled for a 14-bit DAC and utilizes slightly less than the DAC full-scale input range, which is -0.001dBFS to be precise.

```
function ScaledData = ScalePattern(PatIn,N_bits,dBFS)
```

```
% Input variable are:

%PatIn - source data, real values ranging from -1.0 to +1.0

%N_bits - resolution of the output pattern

%dBFS - total output power, relative to full-scale (negative
value)

% Output variable are:

%DecData - output pattern scaled within the range 0 to 2^N - 1

ampscale=power(10,dBFS/20); % convert dBFS to a linear scaling
factor

if ampscale>1.0 % full-scale = 1, do not allow greater values

ampscale=1;

end

scaled_input= PatIn * ampscale; % apply the scaling before
conversion

DigAmp=power(2,N_bits-1); % peak amplitude

ScaledData =(DigAmp)+round(scaled_input*(DigAmp-1)); %offset
binary

MIN_Pat=min(ScaledData);

MAX_Pat=max(ScaledData);

MinStr = ['Minimum Data=',num2str(MIN_Pat)];

MaxStr = ['Maximum Data=' num2str(MAX_Pat)];

disp(MinStr)

disp(MaxStr)
```

Pattern Storage

The function for storing the patterns, StorePattern, requires two input arguments:

- The file name for the new pattern file is PatternFileName and .csv is recommended for the extension.
- The input pattern data, PatternData, output from the ScalePattern function.

The only output is the success indicator, which has a value of 1 if the file is generated successfully and 0 if unsuccessful. An example of the function call to store the example pattern would appear as follows:

```
success=StorePattern('TT525MHz_50MHzSpace_4.6Gsps_14bit.csv',Scal
edData)


function success=StorePattern(PatternFileName,PatternData)
```

```
% Input:
% Filename: name for the file to be written
% data: data vector to be written
% Output:
% Function returns a '1' if the datafile was properly written

success=0;
fid=fopen(filename,'w');
datalen = length(data);
if (mod(datalen/2,16)==0)
   if (fid~= -1) % file was opened successfully
      OddMsb=zeros(datalen/2,1);
      OddLsb=zeros(datalen/2,1);
      EvenMsb=zeros(datalen/2,1);
      EvenLsb=zeros(datalen/2,1);
      OddMsb(1:datalen/2)=bitand(bitshift(data(1:2:datalen),-
      8),255);
      OddLsb(1:datalen/2)=bitand(data(1:2:datalen),255);
      EvenMsb(1:datalen/2)=bitand(bitshift(data(2:2:datalen),-
      8),255);
      EvenLsb(1:datalen/2)=bitand(data(2:2:datalen),255);
      out_data=[OddLsb'; OddMsb'; EvenLsb'; EvenMsb'];
      fprintf(fid,'%d\r\n',(datalen/2));
      fprintf(fid, '%u, %u, %u, %u\r\n',out_data);
      fclose(fid);
   else
      disp(['Can not open output file: ' filename]);
   end
else
   disp(['Data length must be a multiple of 16']);
end
```

VC707 LED Functions

The VC707 includes several user-defined LEDs for indicating status. The MDS system employs several of these LED status indicators for various functions, as shown in **Table 2**.

Table 2. VC707 LED Status Indicators

LED	COLOR	STANDARD OPERATION	DESCRIPTION
0	Green	On	Interface PLL is locked.
1	Green	On	CLK in running.
2	Green	On	Main PLL is locked.
3	Green	Off	FIFO underflow
4	Green	Undefined	N/A
5	Green	On	Data flow is enabled.
6	Green	Undefined	N/A
7	Green	See description.	Parity error indicator (PERR). For the MAX19692/MAX19693, the LED blinks under normal conditions. For the MAX5879/MAX5882, the LED blinks when PERR is high. Press SW7 to reset.

MUXDAC FPGA Command Set

Table 3 lists the commands that can be sent from the PC to VC707 using the VC707 FPGA Commands section of the GUI. The commands are sent to the MicroBlaze microprocessor, which reacts accordingly. When the command is complete, the microprocessor returns an acknowledge ACK# message. Additional information can be returned as well depending on the command. The response is NAK# if an error occurs.

Table 3. MUXDAC FPGA Commands

COMMAND	SYNTAX	DESCRIPTION
help	help	Print help. Use -a flag for all commands. Try \"help -a\".
reg set	reg set <register space> [register name] [value]	Write to control register fields by name. Only the bits of that field within the control register are modified. Arguments: • register space: Zero or more hierarchical register space names • register name: Name of the control register field to set • value: Value to write to the control register field Example: • reg set Sys_5879_CTRL DLLOFF 1 • reg set Sys_5879_CTRL Perr_on 1 The control fields available to be set in the Sys_5970_CTRL register are as follows: • "LOOPBACK" - Bit[2:0] Out • "Ref_Clock_Sel" - Bit[4:3] Connected to LVDS_BUFG_CLK • "Invert_parity" - Bit[5:5] Connected to Mapper • "Perr_on" - Bit[6:6] To LEDs • "Perr_off" - Bit[7:7] To LEDs • "Swap_Sel" - Bit[13:8] Connected to Swapper • "Clock_Feedback" - Bit[14:14] Connected to LVDS_BUFG_CLK • "Clock_Delay" - Bit[19:15] Connected to LVDS_BUFG_CLK • "Clock_Delay_Enable" - Bit[20:20] Connected to LVDS_BUFG_CLK • "DataMapping" - Bit[24:21] Not Connected • "DCLKRST" - Bit[25:25] Out as 'DCLKRST' • "TxRxReset" - Bit[26:26] Out • "Go" - Bit[27:27] Connected as 'go' to MSYNC • "DLLOFF" - Bit[28:28] Out as 'DLLOFF_CNTRL' • "OutLaneSel" - Bit[30:29] Connected to Mapper/Width Converter • "WidthSel" - Bit[31:31] Connected to Mapper/Width Converter
reg read	reg read [address]	Read a register result. Arguments: • address: 32-bit read address in any format that strtoul parses Response: [read_data] ACK# • read_data: 32-bit read data from the read register location (hex format) Example: • reg read 0x80000000

COMMAND	SYNTAX	DESCRIPTION
reg write	reg write [address] [data]	Write a value to a register. Arguments: • address: 32-bit write address in any format that strtoul parses • data:32-bit write data in any format that strtoul parses Example: • reg write 0x80000000 0xA5A5A5A5
play	play buffer [address] [number of bytes] play dumpregs play start play stop play reset	Configure, start, and stop the play/TX DMA channel. • play buffer [address] [number of bytes]: Configure the base address and length of play buffer. • play dumpregs: Print the play channel registers. Output can only be seen on the Xilinx SDK terminal after it is configured for serial connection. • play start: Start the play channel. • play stop: Stop the play channel. • play reset: Reset play channel of the DMA engine. Arguments: • address: 32-bit starting address in any format that strtoul parses. • number of bytes: How many bytes to checksum. Example: • play buffer 0x80000000 524288
checksum	checksum [address] [number of bytes]	Checksum a region of DDR memory. Command: • checksum [address] [number of bytes] Arguments: • address: 32-bit starting address in any format that strtoul parses • number of bytes: How many bytes to checksum Example: • checksum 0x80000000 256
fill	fill [address] [number of bytes] <word> <-flags>	Fill a range of memory with a pattern. Command: • fill [address] [number of bytes] <word> <-flags> Arguments: • address: 32-bit starting address in any format that strtoul parses. • number of bytes: How many bytes to fill. Must be a multiple of 4. • word: 32-bit word to fill memory with. flags: • --ramp -r: Fill with a 32-bit ramp between start and stop values. • --start -s [arg]: Starting value for fill pattern. Defaults to 0. • --stop -p [arg]: Terminal value for fill pattern. Defaults to 0xFFFFFFFF. • --word -w [arg]: How many bytes per word in ramp. Defaults to 4. Example: • fill 0x80000000 0x20000 0xA5A55A5A • fill 0x80000000 0x20000 --ramp -s 0x1234 --stop 0x5678
ping	ping	Return ACK#.
memmap	memmap	Display memory map address.

COMMAND	SYNTAX	DESCRIPTION
selrefclk	selrefclk [arg]	Debugging assistance and allow on-board clocks to be selected as the reference clock. Command: • selrefclk [arg] Arguments [arg] • 0 - Select the FMC clock (default) • 1 - Select the Si570 on-board programmable oscillator • 3 - Select the SMA usr_clk
setdataswap	setdataswap [arg]	Swap 8, 16, 32, 64, 128, and 256-bit lanes of data. Command: • seldataswap [arg] Arguments [arg]: • 0 - No swapping (default) • 1 - 8-bit swap • 2 - 8-bit swap • 3 - 8-bit swap • 4 - 32-bit swap • 8 - 32-bit swap • 12 - 32-bit swap • 16 - 128-bit swap • 32 - 128-bit swap • 48 - 128-bit swap Example: • setdataswap 0 (FPGA programmed value for no swapping)
setwidthmode	setwidthmode [arg]	Select between 2 and 4-lane output widths. Command: • setwidthmode [arg] Arguments [arg] • 1 - 2:1MUX mode, 256-bit channel AC configured output • 2 - 2:1MUX mode, 256-bit channel BC configured output • 4/5/6/7 - 4:1MUX mode, 512-bit channel configured output Example: • setwidthmode 7 (FPGA programmed value for 4:1MUX mode)
setbankdelay	setbankdelay [banknumber] [delay value]	Set the ODELAY value for all the outputs in a given IO bank. Command: • setbankdelay [bank number] [delay value] Arguments [bank number]: 0 to 3 • 0 - IO Bank 19 Data lane A FMC LA00 through LA16 • 1 - IO Bank 34 Data lane B FMC LA17 through LA33 • 2 - IO Bank 36 Data lane C FMC HB00 through HB21 • 3 - IO Bank 35 Data lane D FMC HA00 through HA23 Arguments [delay value]: 0 to 31, default value is 0 for bank 0 and bank 1, 1 for bank 2, and 2 for bank 3 • setbankdelay 0 0 • setbankdelay 1 0 • setbankdelay 2 1 • setbankdelay 3 2 This delay value indicates the ODELAY delay count. Each tap increments by a delay of 78ps. The values are set to the default value of each bank on start-up.
setclockfeedback	setclockfeedback [arg]	Enable the external clock feedback loop. Command: • setclockfeedback [arg] Arguments [arg] • 0 - External clock feedback loop disable (default). • 1 – External clock feedback loop enable.

COMMAND	SYNTAX	DESCRIPTION
setclockdelay	setclockdelay [delay value]	Set the clock delay count in the ODELAY component in the clock feedback loop. Command: • setclockdelay [delay value] Arguments [delay value]: 0 to 31, default = 0 This delay value indicates the ODELAY delay count. Each tap increments by a delay of 78ps in the clock feedback loop. This value is set to 0 by default.
quit	Quit	Quit the program.

Automating RF DAC Measurements with the MDS GUI

This section describes the automation interface within the MDS GUI, the supported commands and their syntax, and the requirements for enabling and connecting to the automation interface.

Note: This section does not define how the command structure or response values are used to achieve any specific functionality.

The reader can configure and control the MDS from an external, customer supplied, automation application (client) using the IC and the EV kit data sheets for the RF DAC under test. The automation interface disables the local control of the GUI and starts a TCP/IP server. Commands are received and processed through the user-defined port on the local host. The server continues to process any connected client until it receives a TCPIP_EXIT command, at which point local control of the interface is restored.

Command Syntax

The command syntax is as follows:

[ret_string] = BOLD_CAPS_ITALIC{?} {argument value}

Each object in the command is defined as follows:

- [ret_string]: The values defined by the command are returned in a string format. The string must be decoded for the specific value type which might include one or more boolean, integer, double, or string types.
- BOLD_CAPS_ITALIC: Command to be executed.
- {?}: (Optional) Indicates that the command is a query and returns the parameter of interest.
- {argument value}: Values to use with the command and might include one or more boolean, integer, double, or string types.

Examples

To assert the HARDWARE MUTE signal and return TRUE if successful, use the following command:

MuteState = HARDMUTE TRUE

To return the list of available frequencies to the double ListVar, use the following command:

ListVar = FCLK_LIST?

Note: The return value from the automation server is in string format and must be processed accordingly within the external controller's environment.

Accessing the Service

The GUI application opens and listens to a TCP/IP port on the local host machine when enabled. An external application accesses this port by connecting to the machine address/input port, such as: 168.0.0.1/2055

External Command Listing

Table 4 lists the external command definitions.

Table 4. External Command Definitions

COMMAND	DEFINITION
ENABLEIMPACT	Notify the software that the IMPACT tool is installed.
CFGFPGA {CfgFileName.bit}	Download the FPGA configuration to the VC707.
PING	Send a PING to the VC707. This command returns a [string]. • ACK# is an acknowledge. • NACK# is a not acknowledge.
STOPPATTERN	Stop looping the current pattern. This command returns [TRUE/FALSE]. • TRUE = The stop executed normally. • FALSE = The command failed.
STARTPATTERN	Start looping the currently selected pattern. This command returns [TRUE/FALSE]. • TRUE = The start executed normally. • FALSE = The command failed.
SELECTPATTERN{?} {integer PATTERNLIST_INDEX}	Select the pattern to use from the current list of available patterns. This command returns [TRUE/FALSE, string PATTERN_NAME] • TRUE = Valid index selected. • FALSE = The command failed. • PATTERNLIST_INDEX = The index of the desired test pattern in the current list.
LOADPATTERNS{?}	Load the pattern files listed in PatListFile.txt and transfer to the VC707 memory This command returns [TRUE/FALSE, STRING PATTERNLIST] TRUE = Valid index selected. FALSE = The command failed. PATTERNLIST = Listing by name of the patterns that successfully load.
RTL	Return to Local: Return control of the application to the GUI interface. This command returns a [string]. • ACK# is an acknowledge. • NACK# is a not acknowledge.

MDS Installation Instructions

The MDS system consists of the Maxim supplied software, FPGA firmware, and the MUXDAC EV kit hardware with the Xilinx Virtex-7 FPGA evaluation board. The software installation is a five-step process.

1. MDS GUI Installation
2. .NET Framework 4 Installation (not required for Windows 10 systems)
3. Xilinx ISE 14.7 Lab Tools Installation
4. Hardware Setup
5. USB 2.0 Driver Installation

Step 1 – Install the MDS GUI

The MDS GUI is installed using an executable that can be downloaded from the Maxim website (**www.maximintegrated.com**). Use the search term MUXDAC to locate the landing page for the MDS and follow the link to download the software. A software license agreement must be acknowledged to complete the download.

Start installation by executing the downloaded file. The installation program provides you with several options. The default options are recommended for this installation.

Note: The MDS GUI interacts with the Xilinx Impact tool (installed in **Step 3 - Download and Install Xilinx ISE 14.7 Lab Tools**) using a Command Line call that includes the full path name of the desired programming file (.bit or .mcs). If this path or the path back to the Impact tool contains any spaces, the Impact tool aborts execution.

The step-by-step process for installing the MDS GUI is as follows:

1. Download the MUXDAC Installer program from the Maxim website.
2. Run the installer.

   Note: The operating system can require the user to approve the installation. If so, this occurs prior to the next steps in this procedure.
3. Change the Installation Folder if desired (no spaces). Click **Next**.

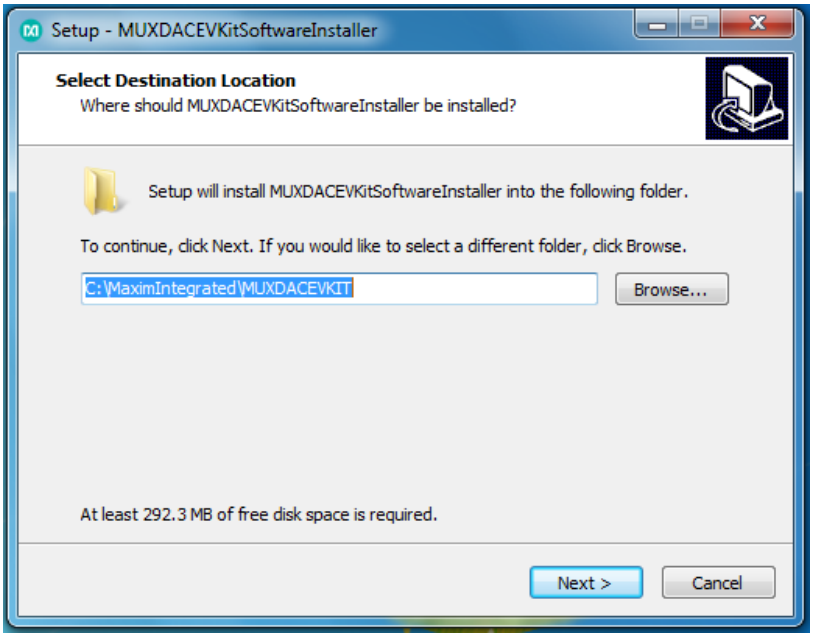


Figure 12. Select the MUXDAC installation location.

4. Select options for adding shortcuts (defaults are desktop and start menu). Click **Next**.

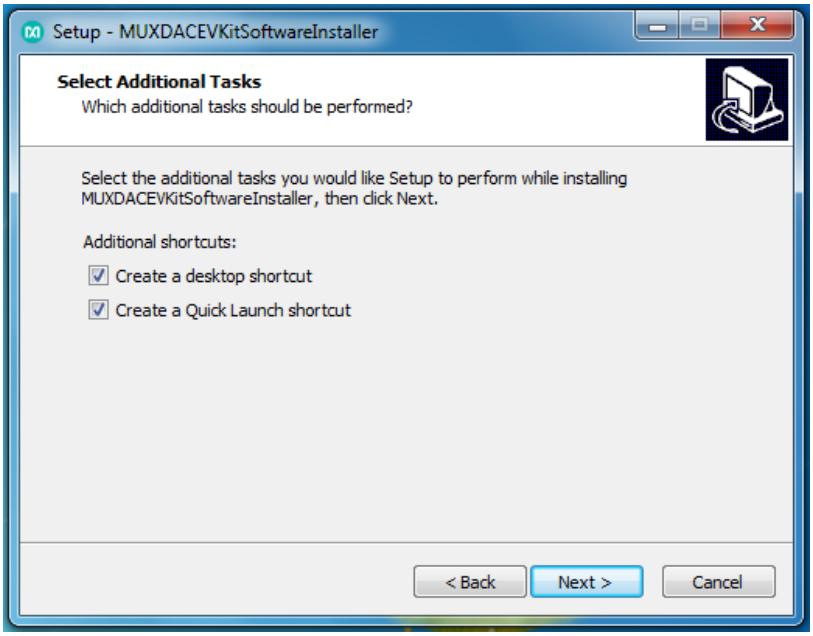


Figure 13. Add MDS GUI shortcuts.

5. Review installation settings and click **Install** to accept and install.

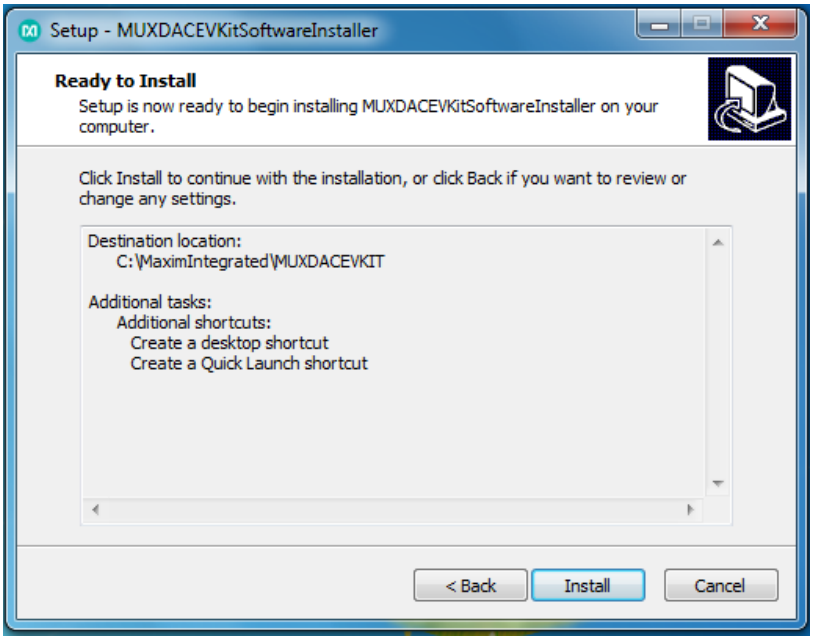


Figure 14. Proceed with MDS GUI installation.

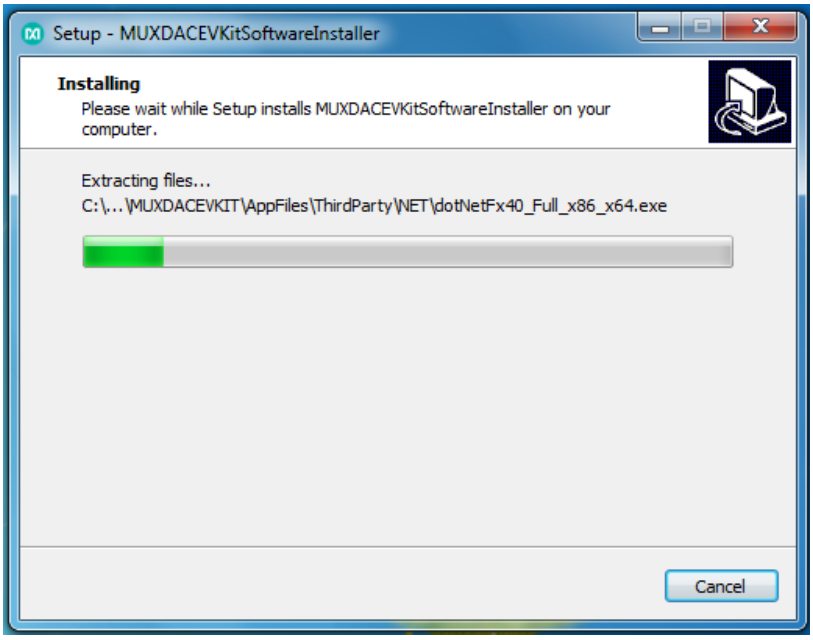


Figure 15. MDS GUI installation in progress.

6. Click **Finish** after completion.

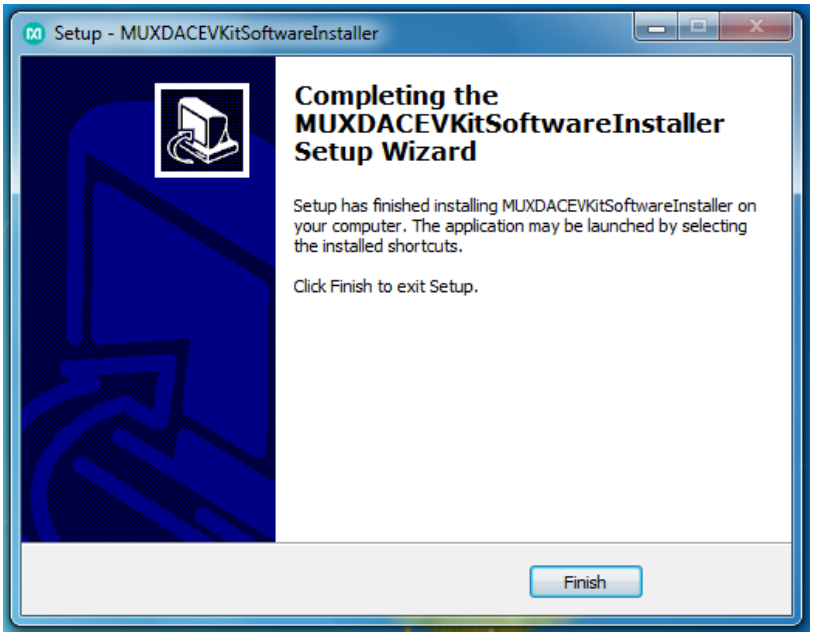


Figure 16. MDS GUI installation complete.

7. Proceed to ***Step 2 - Install .NET Framework 4*** section.

Several files and folders are added to the PC during the installation of the MDS GUI. **Table 5** lists the general descriptions of the installed files and folders.

Table 5. Installed Files and Folders

FILE OR FOLDER	DECRIPTION
MUXDACEVKITSoftwareController.exe	Application program.
AppFiles	Directory application support files including the USB_MS_Bulk_Transfer driver and .NET Framework 4 installation package (required for Windows 7 systems).
Documentation	Directory for storing device specific documents. This folder is installed with only a readme.txt file.
PatternFiles	Directory with sample pattern files and MATLAB routines for generating additional CW patterns.
VC707Files	Directory with FPGA programming files.
Miscellaneous DLLs and an uninstall executable	Supporting DLL files for software operation and uninstalling the application.

Step 2 – Install .NET Framework 4

The MDS GUI has features that require the .NET Framework 4 environment. These features are natively supported in the Windows 10 operating system but must be added to Windows 7. Maxim includes the .NET Framework 4 installation package in the MDS GUI installer. Alternatively, the web installer is available directly from Microsoft (**www.microsoft.com**).

Locate the .NET Installer package in the MUXDACEVKIT\AppFiles\ThirdParty\NET folder. The file name is dotNetFx40_Full_x86_x64 with the .exe extension. Double click the file name to launch the installation program.

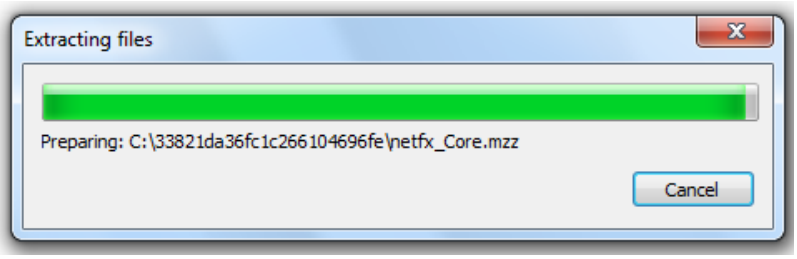


Figure 17. Extract .NET Framework 4 components.

The step-by-step process for installing the .NET Framework 4 is as follows:

1. Using the File Browser, open the \AppFiles\ThirdParty\NET folder.
2. Double click on the dotNetFx40_Full_x86_x64.exe executable file.
3. If the systems asks, click **Yes** to allow the program to execute.
4. Click the checkbox to accept the license terms.

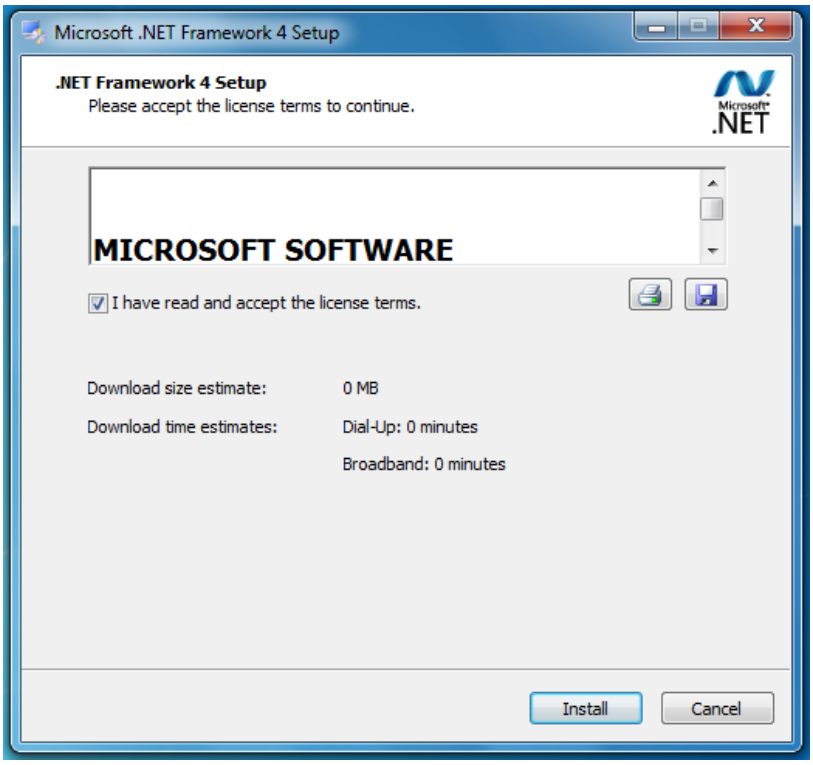


Figure 18. Accept .NET Framework 4 license agreement.

5. Click **Install** and wait for completion.

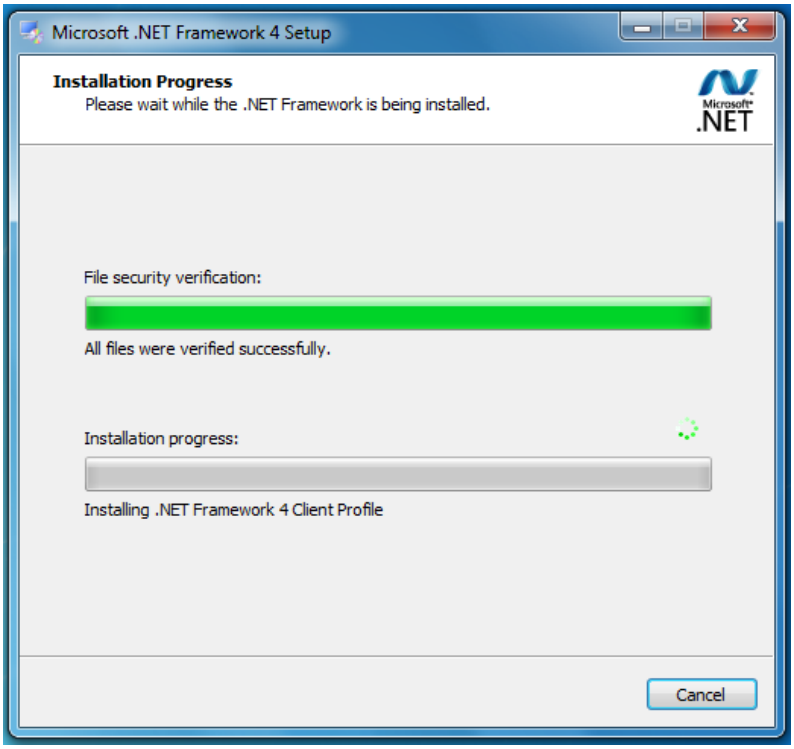


Figure 19. .NET Framework 4 installation in progress.

6. Click **Finish**.

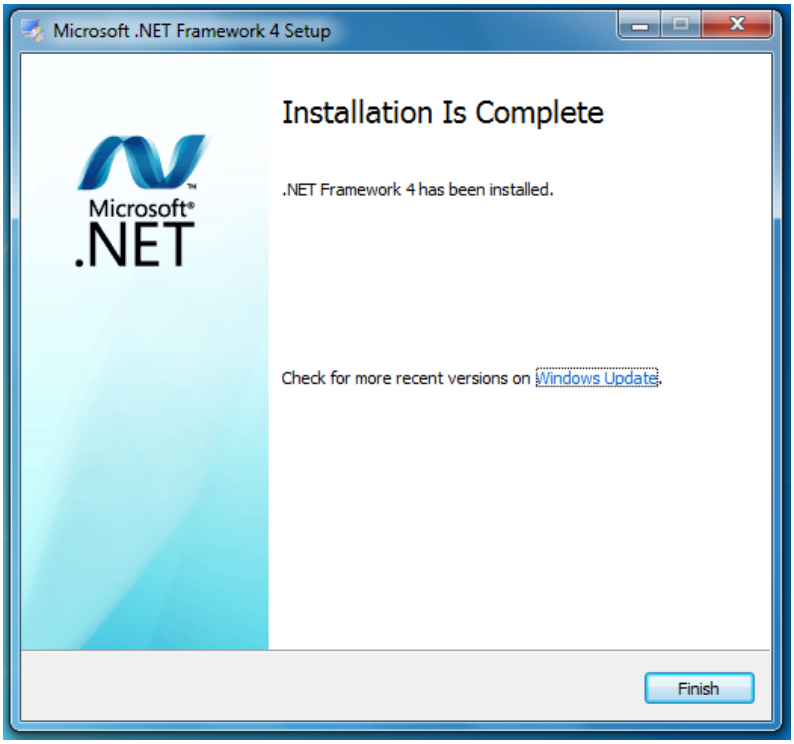


Figure 20. .NET Framework installation complete.

7. Proceed to the ***Step 3 – Download and Install Xilinx ISE 14.7 Lab Tools*** section.

Step 3 – Download and Install Xilinx ISE 14.7 Lab Tools

The Xilinx ISE 14.7 Lab Tools is available to download for free on the Xilinx website (**www.xilinx.com**), as shown in **Figure 21**. Click the All Platforms link and follow the process for completing the download. When the download is complete, unzip the folder to proceed with installation.

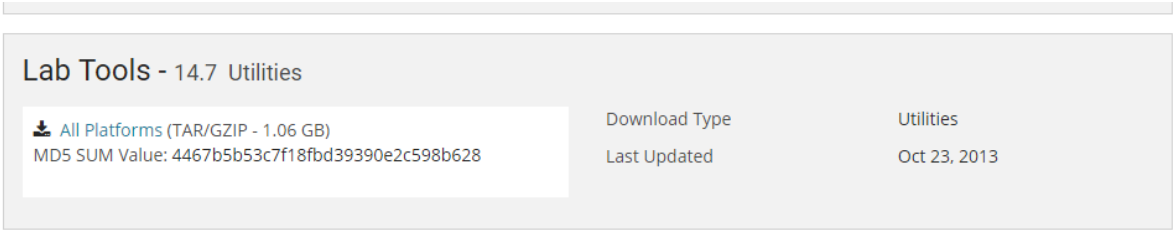


Figure 21. Xilinx ISE 14.7 Lab Tools download.

Open the unzipped folder and double click the xsetup.exe file to launch the installation application. The step-by-step process for installing the Xilinx ISE 14.7 Lab Tools is as follows:

1. The welcome screen appears. Click **Next** to continue.

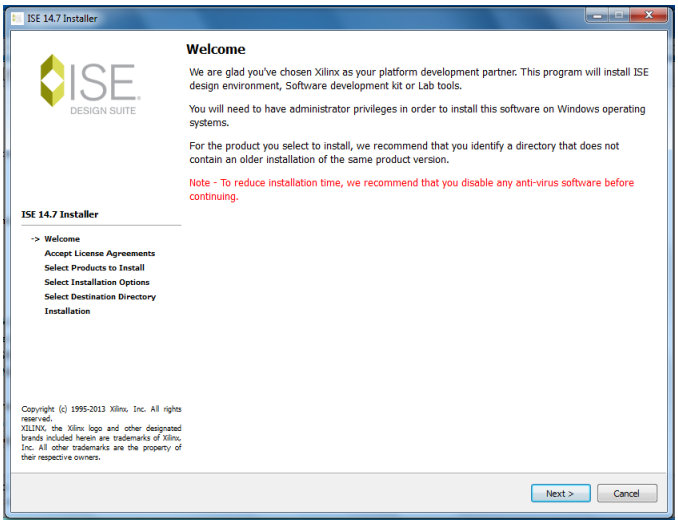


Figure 22. Starting the Xilinx ISE 14.7 Lab Tools installation program.

2. Click the checkbox next to “I accept and agree to the terms and conditions above.”
3. Click the checkbox next to “I also accept and agree to the following terms and conditions.”
4. Click the **Next** button to proceed.

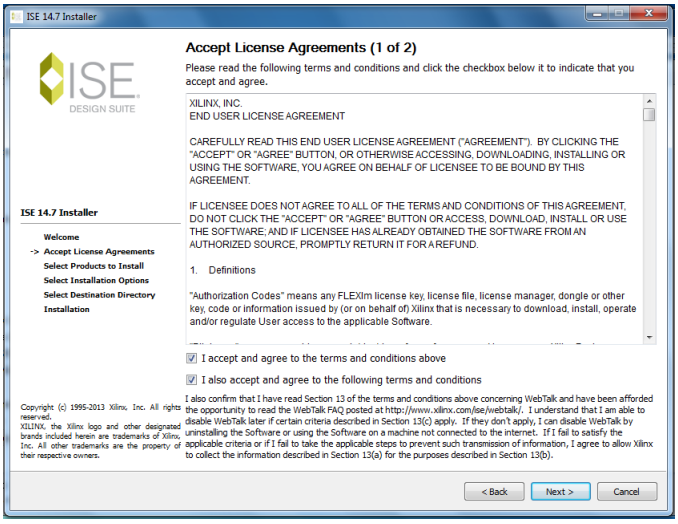


Figure 23. Accepting terms and conditions.

5. Click the checkbox next to “I accept and agree to the terms and conditions above.”
6. Click the **Next** button to proceed.

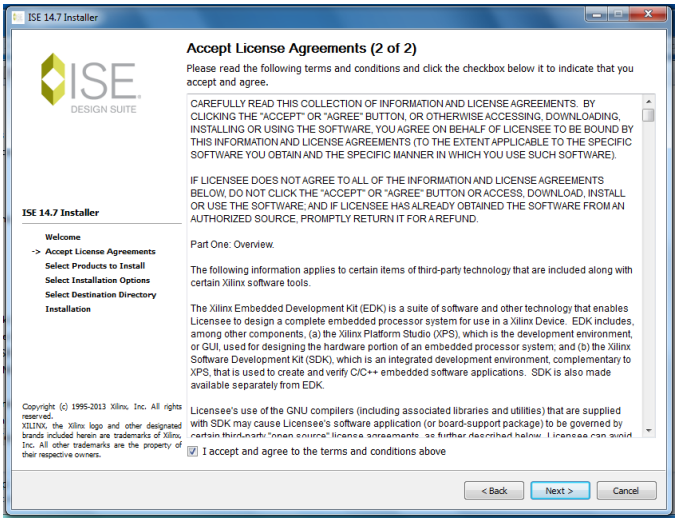


Figure 24. Accepting additional terms and conditions.

7. Lab Tools – Standalone Installation is the only option. Click **Next** to proceed.

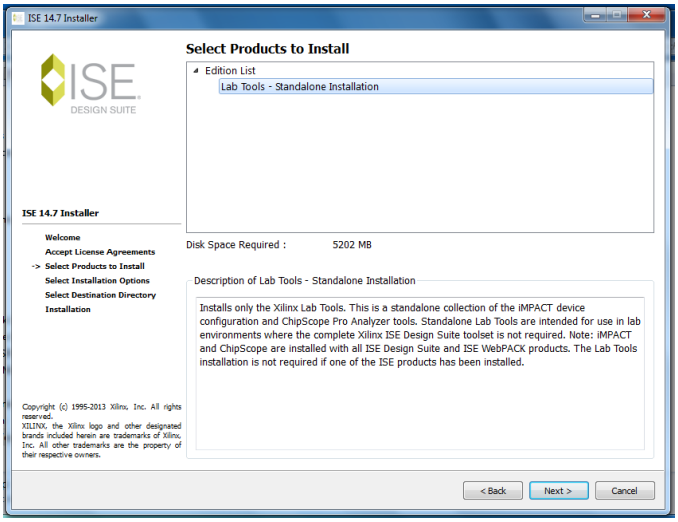


Figure 25. Products to install.

8. Unselect the “Acquire or Manage a License Key” (no license is required).
9. Unselect the “WebTalk” option if desired (recommended for this application).
10. Click the **Next** button to proceed.

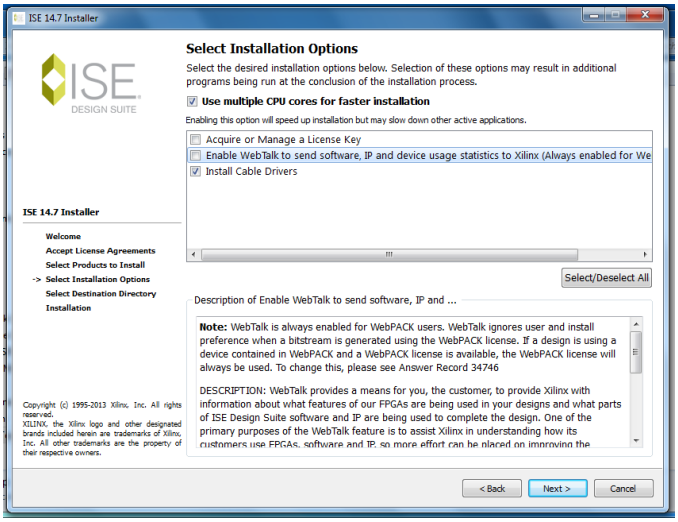


Figure 26. Installation options.

11. Click **Next** to use the default settings on the Select Destination Directory page.

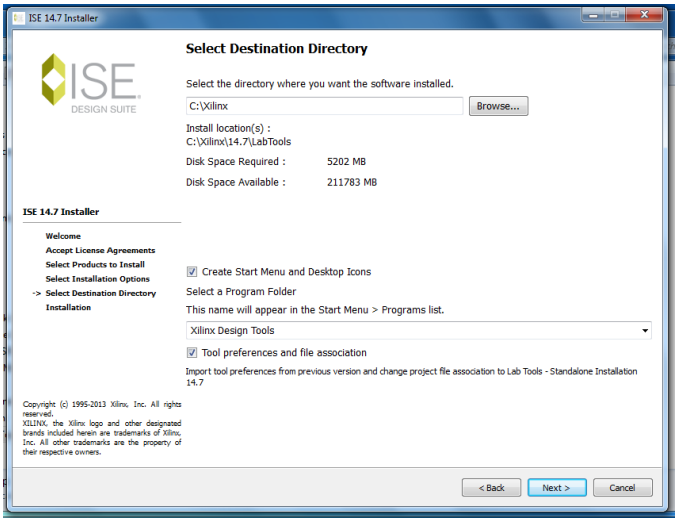


Figure 27. Select the destination folder, shortcuts, and tool preferences.

12. Review the installation settings and click **Install**.

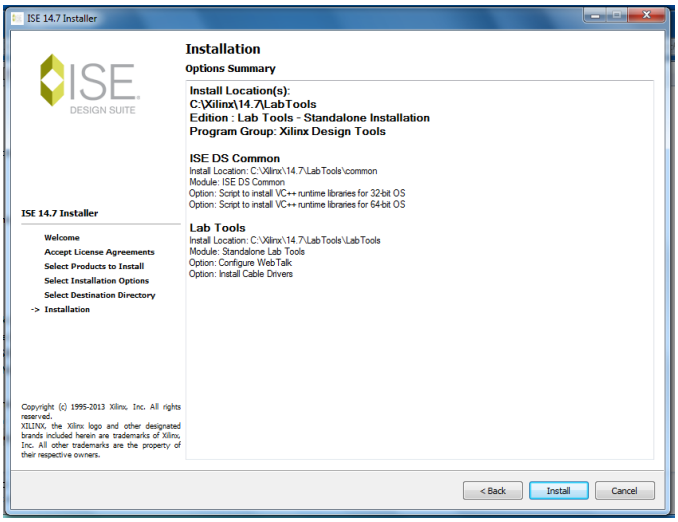


Figure 28. Review installation and install.

Installation can take up to 30 minutes to complete.

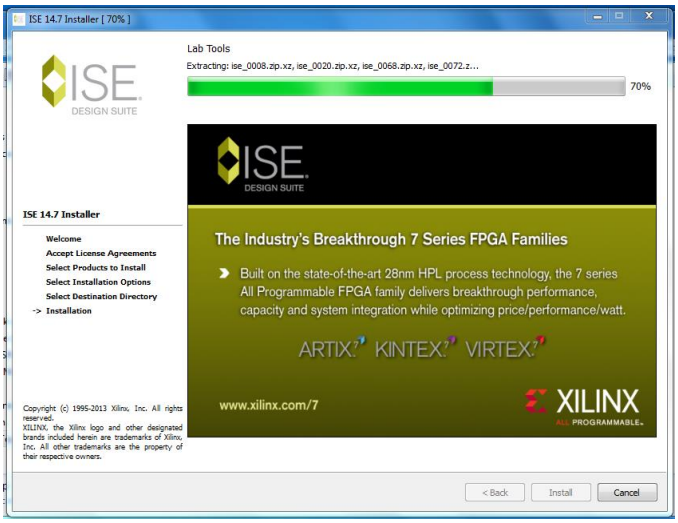


Figure 29. Installation begins with extraction of the components.

13. The installation includes the Microsoft Visual C++ 2008 Redistributable Setup. When prompted, click **Next** to run.

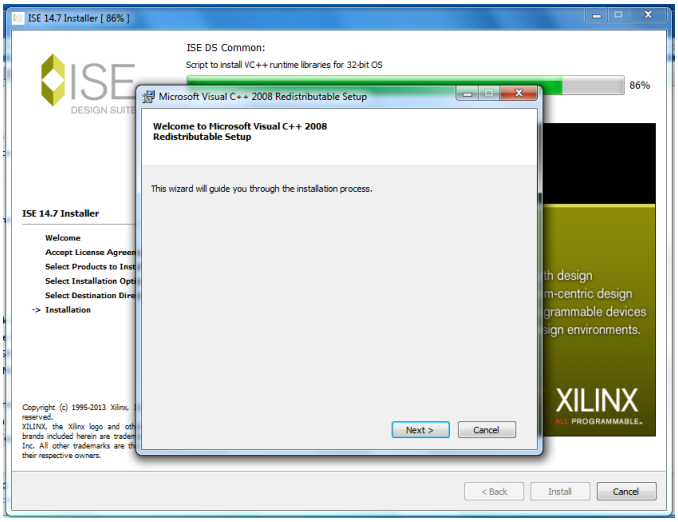


Figure 30. Prompt to run Microsoft Visual C++ 2008 Redistributable setup.

14. Click the checkbox to accept the license terms, and then click **Install** to proceed.

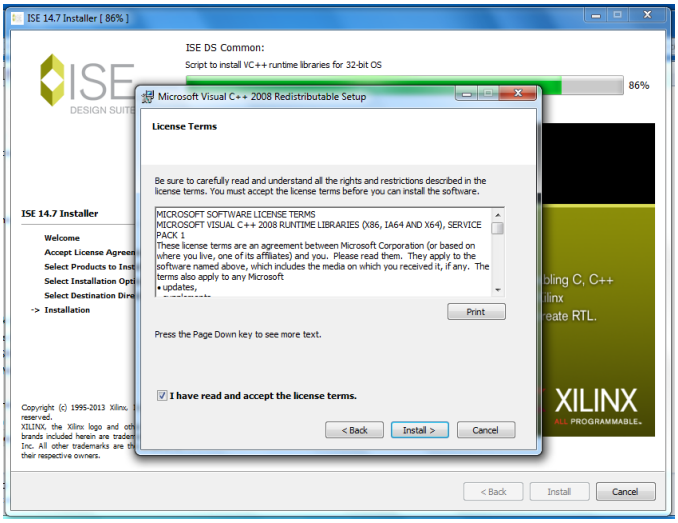


Figure 31. Accept the license and install.

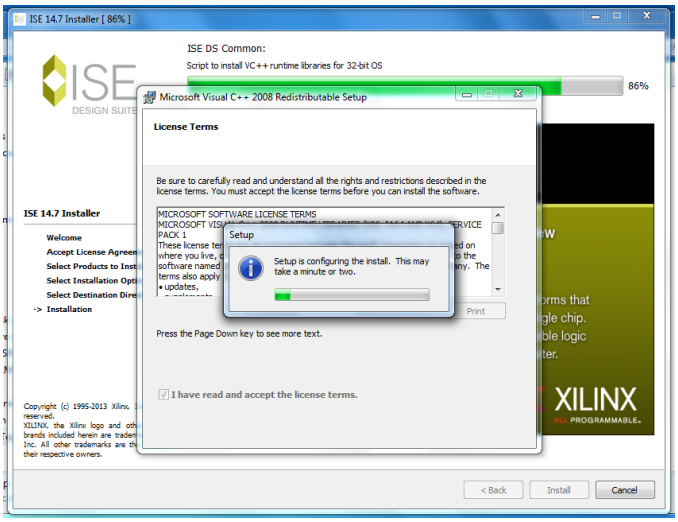


Figure 31. Configuring the Microsoft Visual C++ 2008 Redistributable setup.

15. Click **Finish** when prompted.

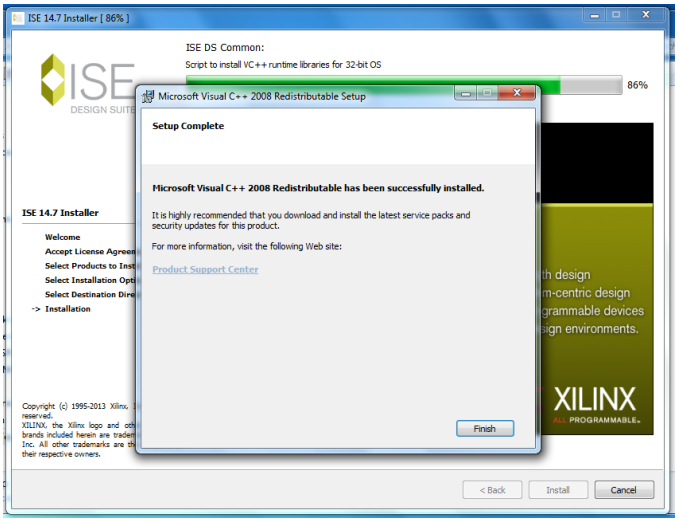


Figure 33. Microsoft Visual C++ 2008 Redistributable setup is complete.

16. The Xilinx Installer also requires the Microsoft Visual C++ 2008 Redistributable Updates. When prompted, click **Next** to install.

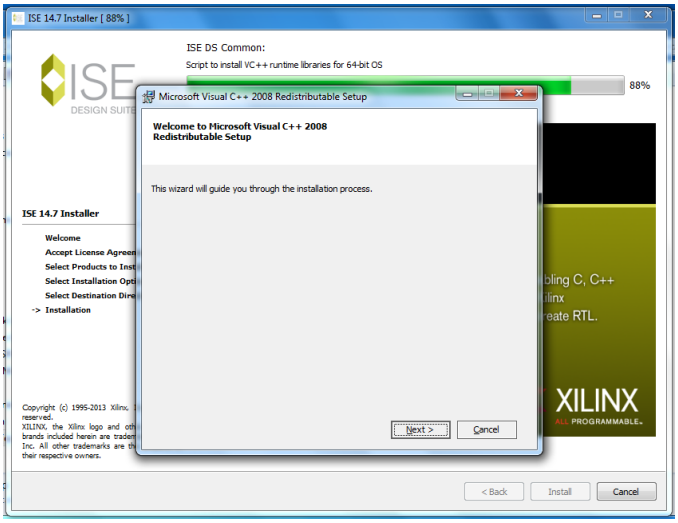


Figure 34. Update required for Microsoft Visual C++ 2008 Redistributable.

17. Click the checkbox to accept the license terms, and then click **Install** to proceed.

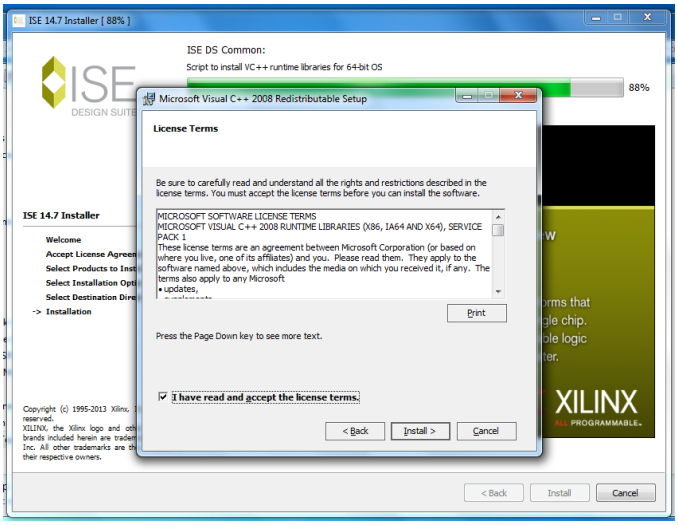


Figure 35. Accept license for update and install.

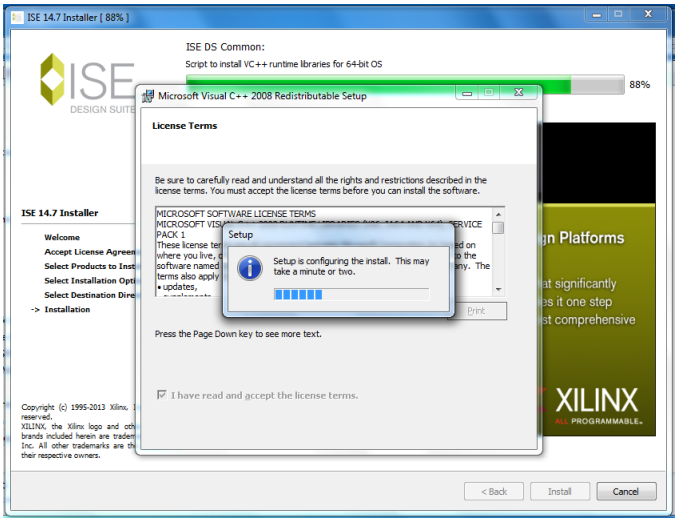


Figure 36. Updating Microsoft Visual C++ 2008 Redistributable.

18. Click **Finish** when prompted.

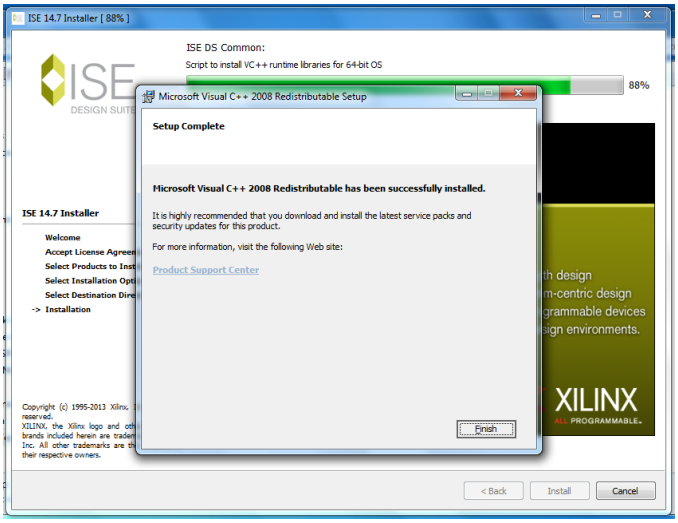


Figure 37. Microsoft Visual C++ 2008 Redistributable Update is complete.

19. Ensure all USB connections to the VC707 are disconnected and that no other Xilinx programming hardware is connected.

20. Click **OK** to continue.

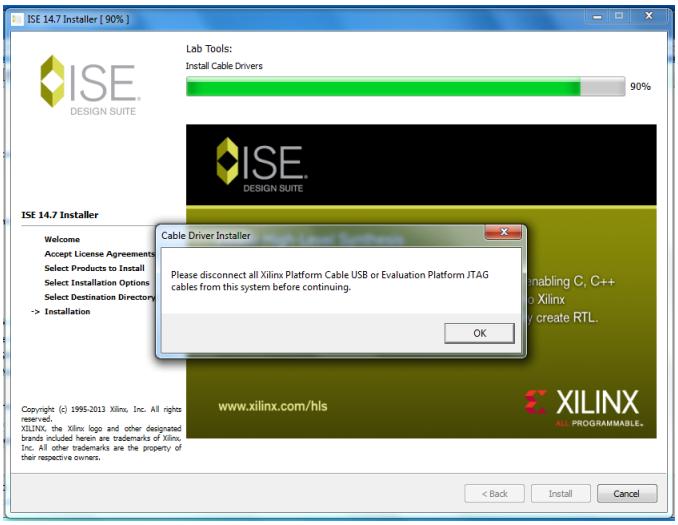


Figure 38. Prompt to disconnect all USB cables from Xilinx hardware.

21. Click **Install** to install the Jungo Software.

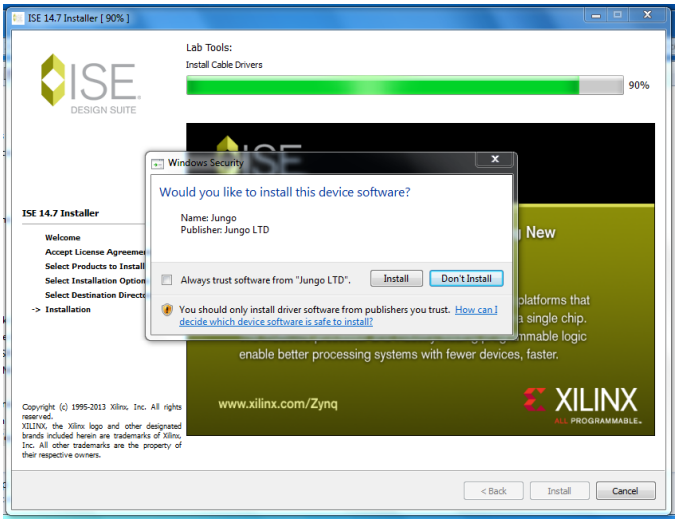


Figure 39. Prompt to install Jungo Software.

22. Click **Install** to install the device software for the Xilinx cable drivers.
23. Wait for the installation to complete.

Figure 40. Cable driver installation.

24. Click **Finish** when prompted.
25. Open a File Browser and navigate to C:\Xilinx\14.7\LabTools\LabTools\bin.

26. Open either the “nt” or “nt64” folder and run the install_drivers.exe program.

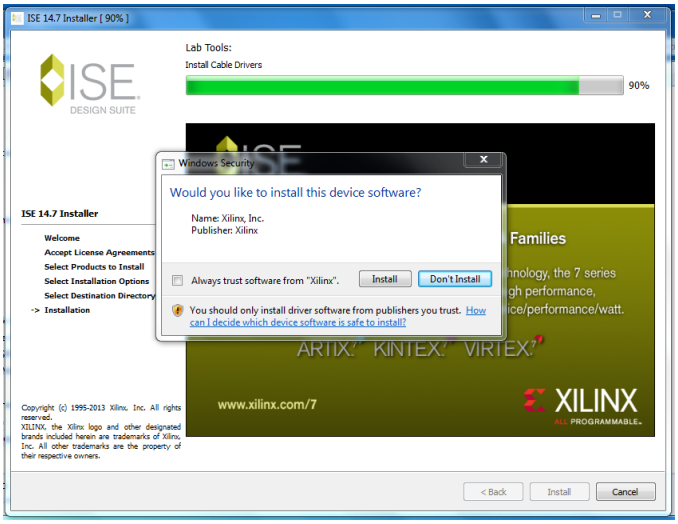


Figure 41. Prompt to install device software (driver installation).

27. A window flashes briefly during execution.

28. The Lab Tools installation is complete. Proceed to the ***Step 4 – Hardware Setup*** section.

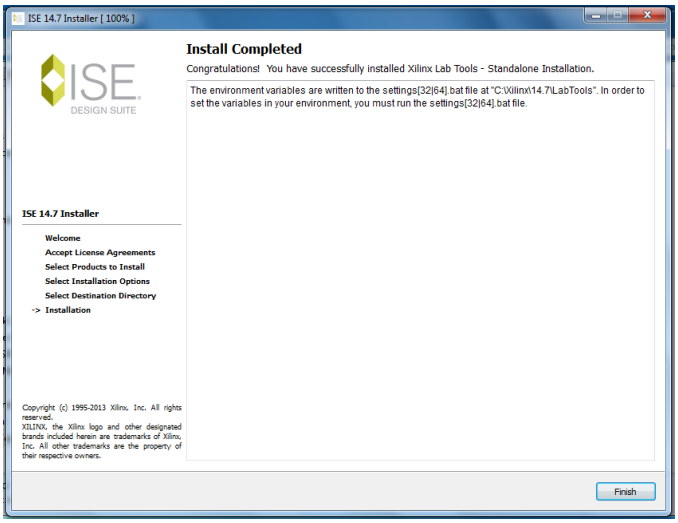


Figure 42. Lab Tools installation is complete.

Step 4 – Hardware Setup

Specific details for setting up the MUXDAC System Hardware are provided in the data sheets for each of the supported EV kits. As shown in **Figure 43**, the step-by-step process is as follows:

1. Connect the DAC clock source to the DAC EV kit.
2. Connect the DAC output to a spectrum analyzer.
3. Connect the FMC to QSH Adapter card to the DAC EV kit.
4. Connect the FMC adapter to the VC707.
5. Connect the power supplies as needed to the DAC EV kit.
6. Connect the power cube provided with the VC707 EV kit to the FPGA board.
7. Connect the Micro-USB port on the FPGA board to the PC.
8. Connect the Mini-USB port on the FPGA board to the PC.
9. Enable the power supplies to the DAC EV kit.
10. Enable the clock signal to the DAC.
11. Power up the VC707 FPGA board and slide the SW12 to the left.
12. The hardware setup is complete. Proceed to the **Step 5 – Install the Final Driver** section,

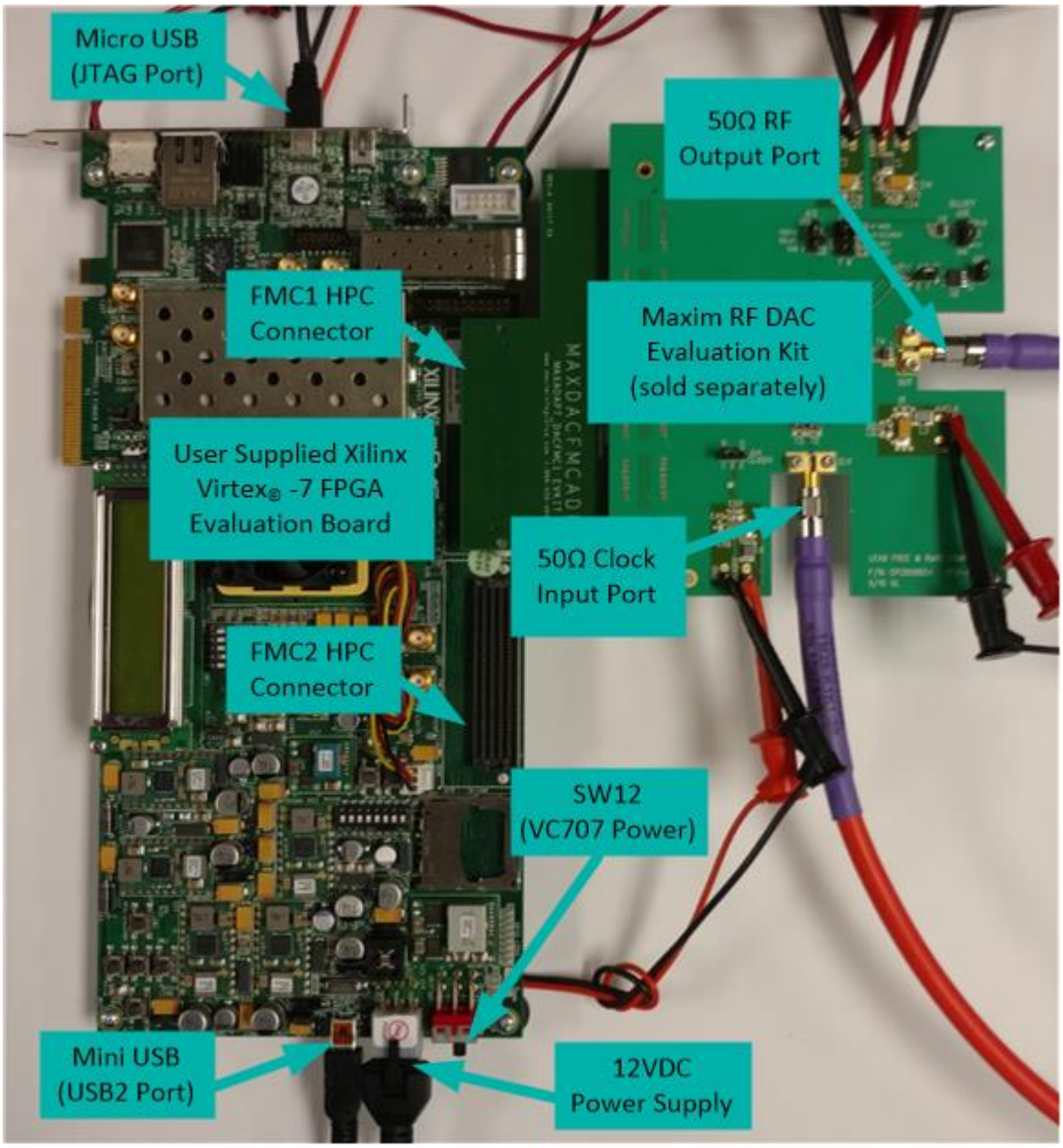


Figure 43. Hardware setup using the VC707 FPGA board and the MAX5882 EV kit.

Step 5 – Install the Final Driver

A special driver is used for the USB 2.0 communication port on the VC707. The ULPI/USB 2.0 port is activated upon completion of the FPGA programming. The PC initially recognizes the newly activated USB 2.0 port as a USB mass storage device and automatically installs the device driver. The USB connection can be viewed in the Device Manager, which indicates an issue by adding the ⚠ symbol to a USB connection, as shown in **Figure 44**. Installation of the correct driver requires that the FPGA is programmed with the MUXDAC firmware and the USB 2.0 port is connected to the PC.

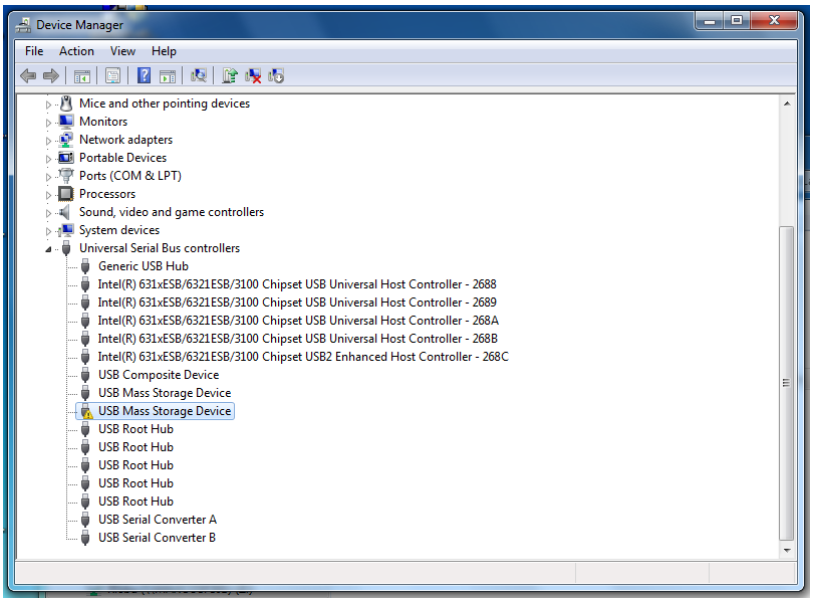


Figure 44. Device Manager showing the USB 2.0 port with the incorrect driver.

The step-by-step process for finalizing the MUXDAC installation is as follows:

1. Launch the MDS GUI (C:\MaximIntegrated\MUXDACEVKIT\MUXDACEVKITSoftwareController.exe).

Figure 45. Splash screen displayed at start-up of the MDS GUI.

2. Wait for the splash screen to close and the main GUI window appears.

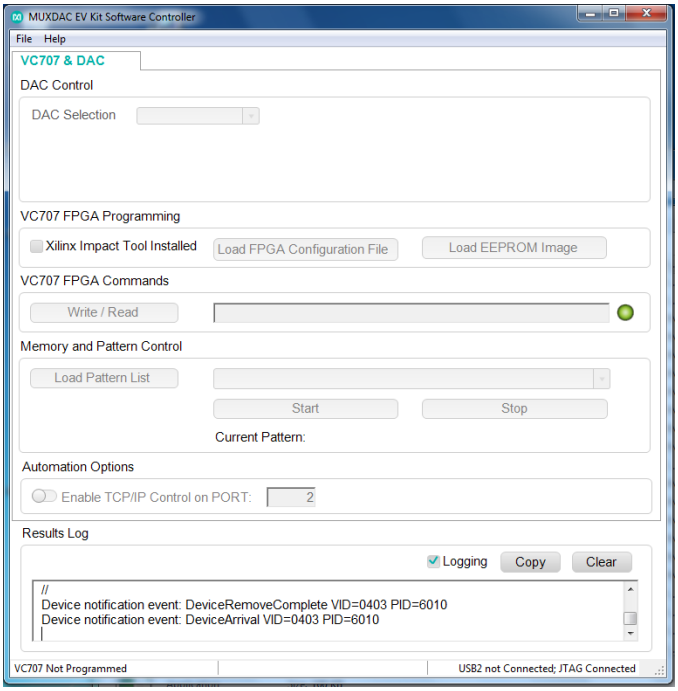


Figure 46. Main GUI window at startup.

3. Verify that the status bar indicates JTAG Connected in the bottom right.

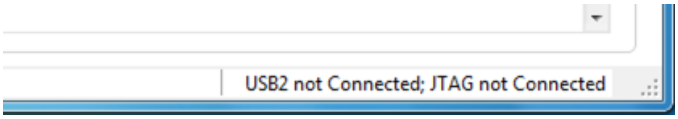


Figure 47. USB 2.0 and JTAG cables not connected. JTAG is required to continue.

4. Reinstall Xilinx Cable Drivers if needed to establish JTAG connectivity. (C:\Xilinx\14.7\LabTools\LabTools\bin\{nt64||nt}\install_drivers.exe)

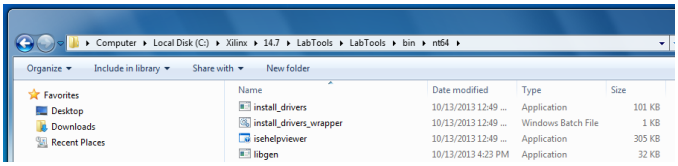
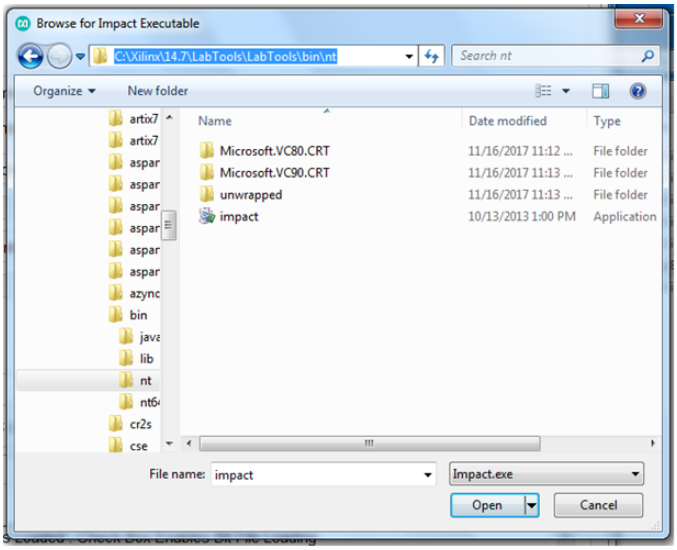


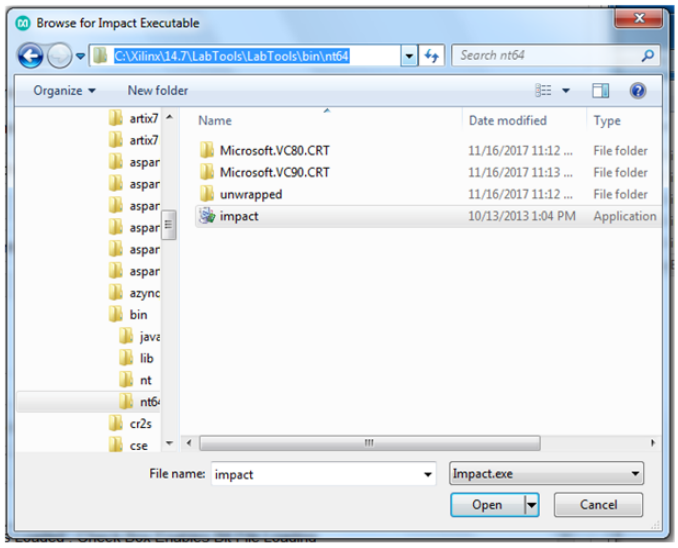
Figure 48. Reinstall Xilinx cable drivers to resolve connection issues.

5. In the VC707 FPGA Programming section, click the checkbox next to Xilinx Impact Tool Installed.

6. When the File Browser appears, navigate to C:\XILINX\14.7\LabTools\LabTools\bin\{nt64||nt}. Select the impact.exe file and click **Open**.



a)



b)

Figure 49. Locate the impact.exe application, 32-bit OS (a) 64-bit OS (b).

7. Click the **Load FPGA Configuration File** button.

8. When the File Browser appears, select the MUXDAC_DSS_vNpN.bit file and click **Open**.
   Note: vNpN is the firmware version.

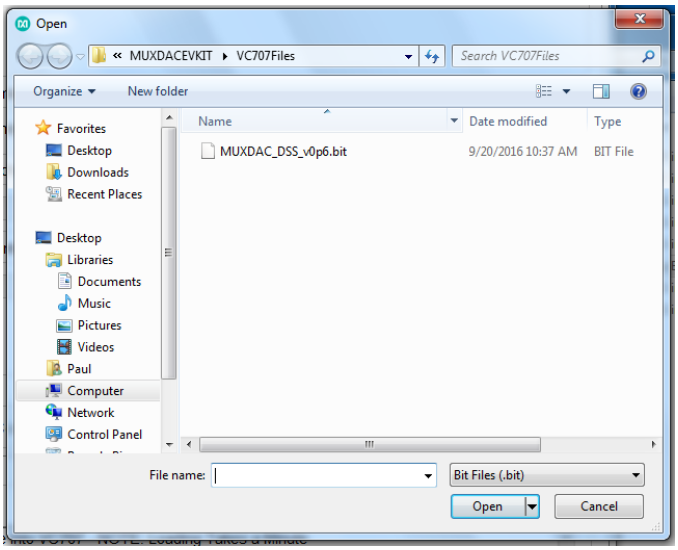


Figure 50. Selecting the FPGA programming file.

9. Wait for the FPGA programming to complete.

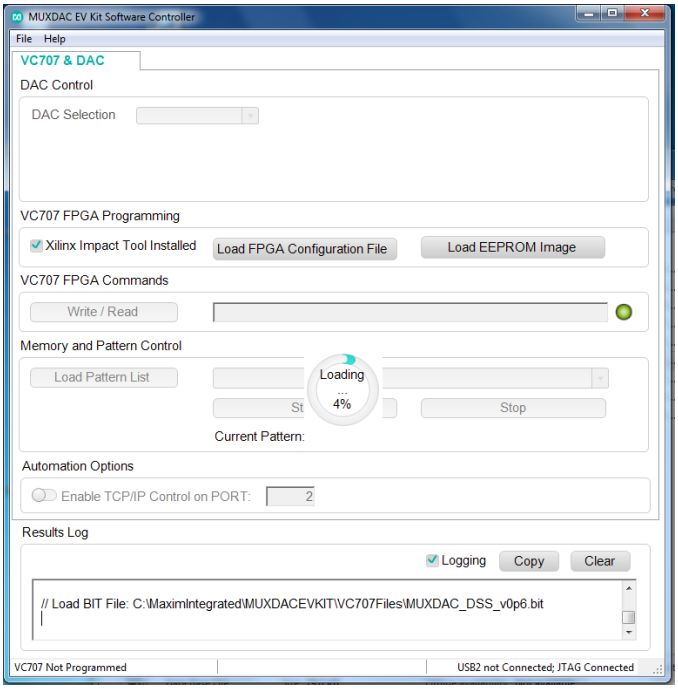


Figure 51. FPGA programming in progress.

10. The GUI throws an error due to the wrong driver installed on the USB 2.0 port.

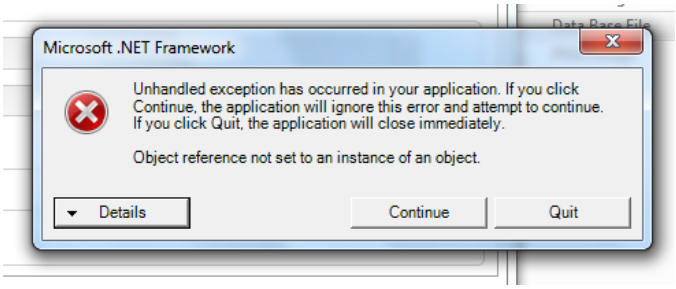


Figure 52. Application error after completing the FPGA programming.

11. Click the **Quit** button to exit the MDS GUI application.
12. Open Device Manager and locate the Mass Storage Device with the ⚠ warning sign.

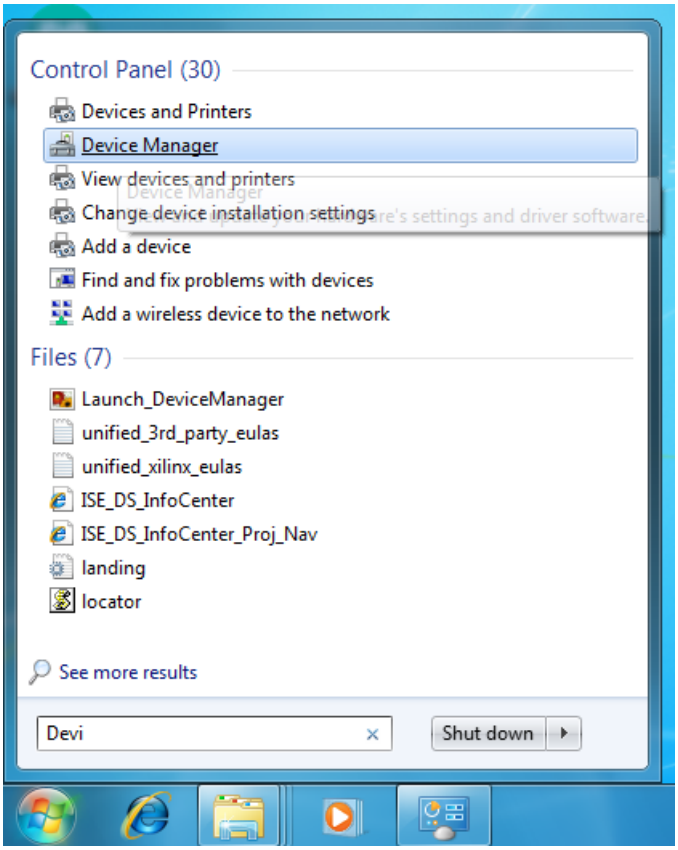


Figure 53. Launching Device Manager.

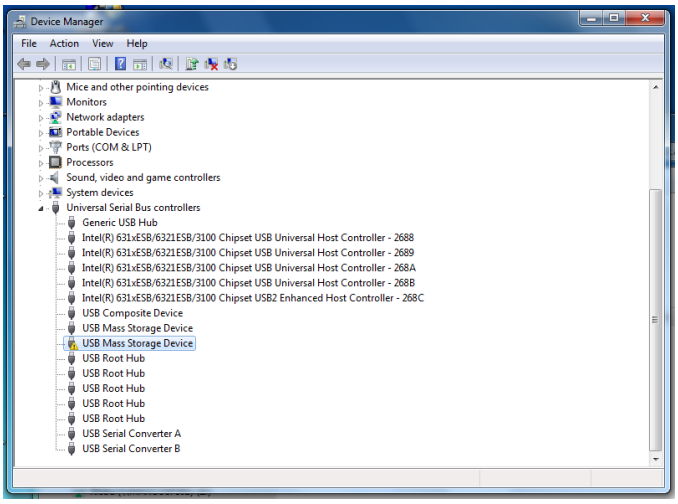


Figure 54. Device Manager with warning on USB Mass Storage Device.

13. Right click the device name and select **Update Driver Software** from the available options.

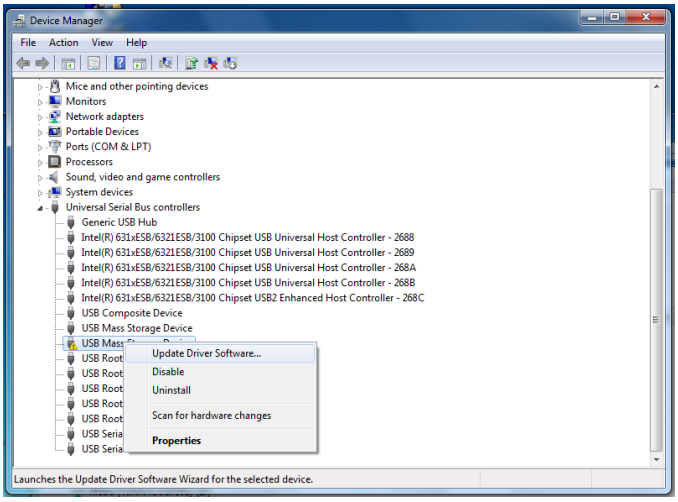


Figure 55. Starting the Update Driver Software process.

14. Click **Browse my computer for driver software**.

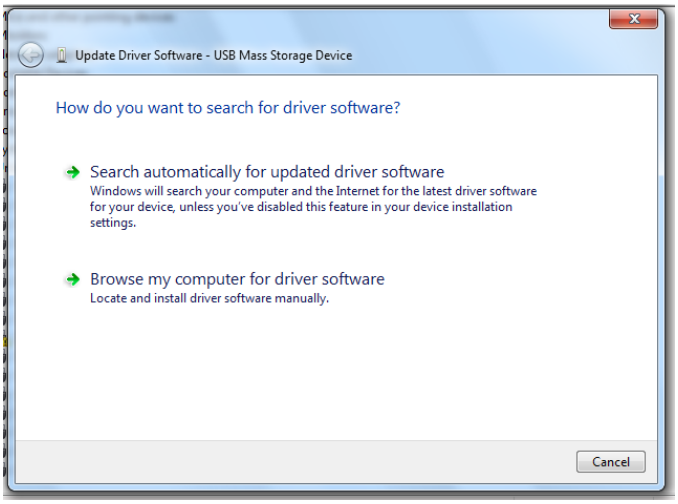


Figure 56. Selecting How to Search for the Driver. Choose Browse my computer for driver software.

15. Click **Let me pick…**.

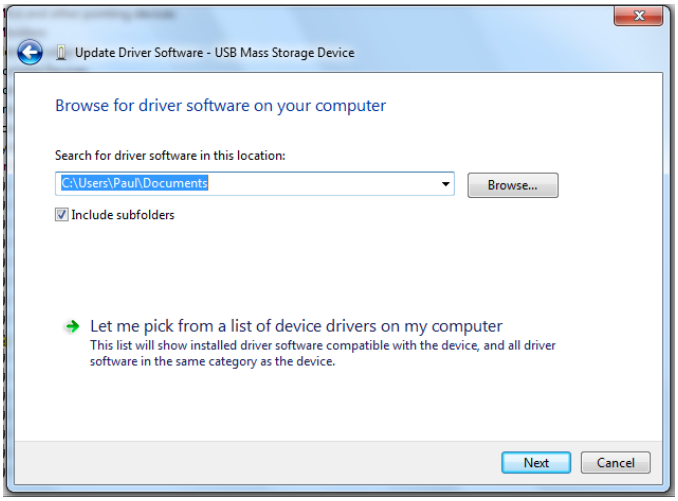


Figure 57. Find the driver. Choose Let me pick from a list….

16. Highlight USB Mass Storage Device in the Model section and click **Have Disk**.

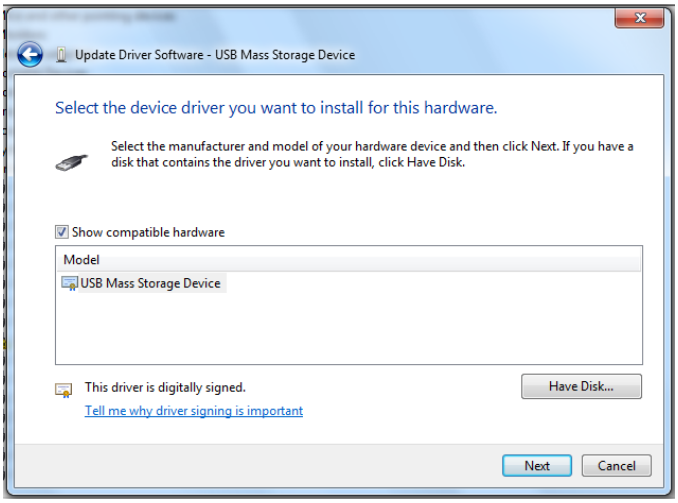


Figure 58. Select the device and click Have Disk….

17. Click **Browse** on the Install From Disk window.

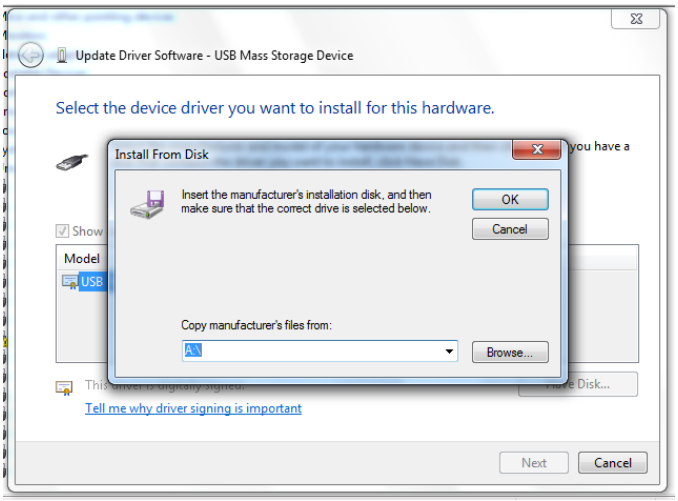


Figure 59. Search for the Driver and click Browse.

18. In the File Browser, navigate to C:\MaximIntegrated\MUXDACEVKIT\AppFiles\ThirdParty\USB_MS_BULK_Transfer.

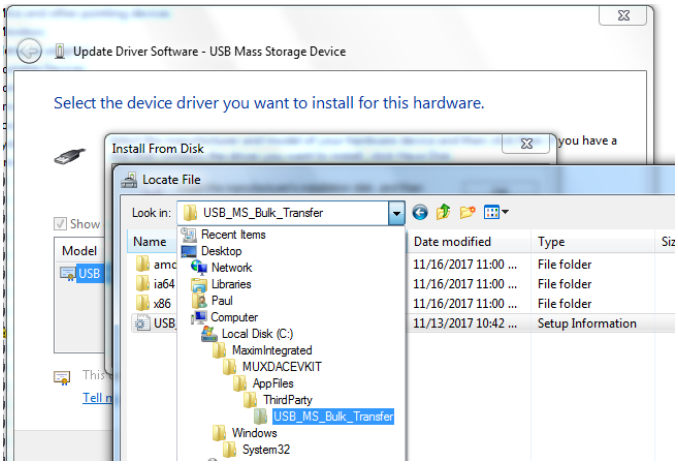


Figure 60. Navigate to the driver file location.

19. Select the USB_MS_Bulk_Transfer file from the list and click **Open**.

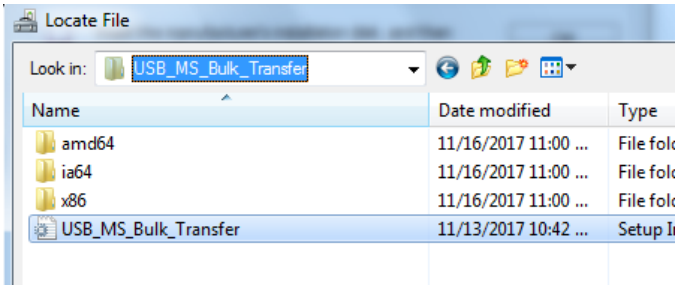


Figure 61. Selecting the driver file.

20. Click **OK** on the Install From Disk window.

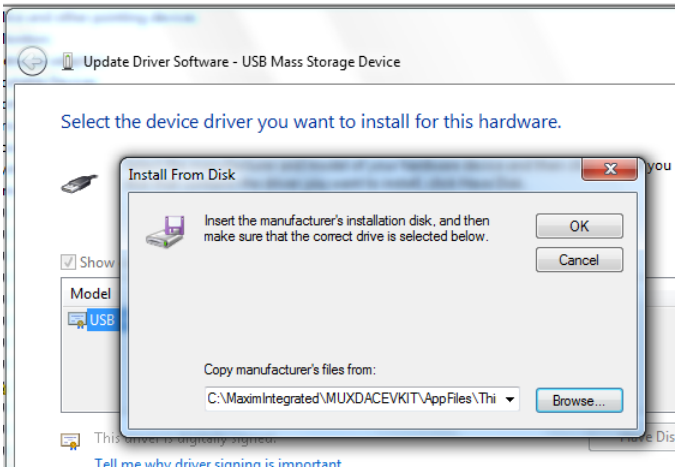


Figure 62. Click OK to install the selected driver.

21. If Windows Security asks for confirmation, click **Install**.

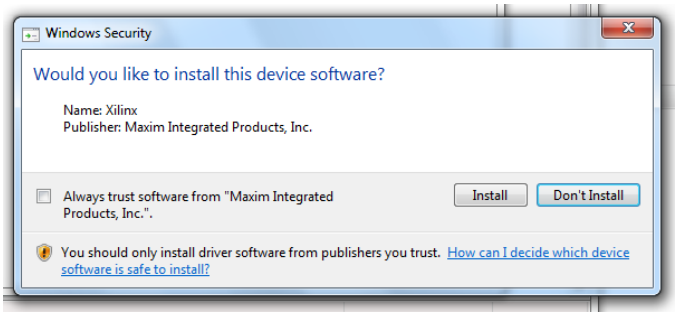


Figure 63. System requests to allow installation. Click Install.

22. Windows typically has an issue with starting this driver under these conditions. A notification window displays indicating the issue.

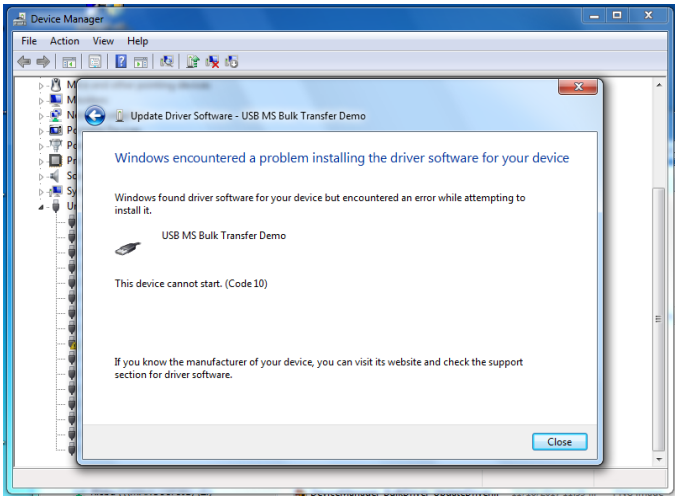


Figure 64. Typical driver installation failure. Reboot is required.

23. Click **Close** to proceed.

24. It is recommended to reboot the PC prior to using the MDS.

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	4/19	Initial release	—