WP477 (v1.0) June 14, 2016

UltraRAM: Breakthrough Embedded Memory Integration on UltraScale+ Devices

UltraRAM is a new memory block in UltraScale+™ families that enables up to 500Mb of total on-chip storage, equating to a 6X increase in on-chip memory vs. 28nm Xilinx FPGAs.

ABSTRACT

Traditional FPGAs and SoCs have included on-chip memory in the form of block RAM and distributed RAM. As devices process more data at greater data rates, the need to buffer or store data close to where it is being processed increases.

New in UltraScale+ families is a larger capacity, flexible memory block called UltraRAM. UltraRAM blocks can be cascaded together to create large on-chip memories.

Using UltraRAM in a design is easy, with Xilinx providing all the tools necessary to incorporate the new, powerful block in a design.

Increasing Demands on Buffering and Storage

UltraScale+™ FPGAs and MPSoCs are capable of transporting and processing vastly more data than previous generation devices. The largest Virtex® UltraScale+ FPGA, the VU13P, contains 128 GTY transceivers, operating at data rates up to 32.75Gb/s, and over 11,000 DSP slices, operating at nearly 900MHz. The resulting 8.4Tb/s of serial bandwidth and 21 TMAC/s of signal processing demand a similar increase in on-chip storage to enable systems to efficiently buffer incoming and outgoing data pre- and post-processing.

Up to and including UltraScale™ FPGAs, block RAM and distributed RAM have been available to serve as quick and efficient on-chip memory. To successfully handle hundreds of megabits of data, however, external memories such as DDR4, DDR3, or RLDRAM3 have traditionally been required.

The UltraScale+ portfolio uses a new storage method. UltraRAM is a large, lightweight memory block that enables UltraScale+ devices to provide in excess of 500Mb of power- and cost-efficient on-chip data storage, equating to a 6X increase in on-chip memory vs. 28nm Xilinx FPGAs.

Flexible cascade capabilities allow UltraRAM blocks to be used in many configurations—from a single block to all blocks in a device connected together. This enables users to create on-chip memory arrays of size to suit their application and eliminate some external memory components, such as QDR SRAM, from the PCB.

UltraRAM Details and Features

For many generations, Xilinx FPGAs and SoCs have used a columnar architecture with columns of the different resources sitting alongside each other. The number of columns and the height of the columns directly determined the size and capability of the FPGA or SoC.

Most Kintex® UltraScale+ and Zynq® UltraScale+ devices contain a single column of UltraRAM, whereas the high-end Virtex UltraScale+ FPGAs contain up to five columns of UltraRAM. UltraRAMs can be used as single 288Kb memories in a design or can be connected together to create larger RAM arrays. All UltraRAMs within the same column can be connected together with dedicated cascade routing for address, data, and control signals. The resulting RAM arrays can be up to 36Mb in Kintex UltraScale+ and Zynq UltraScale+ devices or up to 22.5Mb in Virtex UltraScale+ devices. In the Virtex UltraScale+ family, all the columns of UltraRAM can be connected together using fabric routing to create memory arrays up to 360Mb in the largest device.

Every UltraRAM block is a dual-port synchronous 288Kb RAM with fixed configuration of 4,096 deep and 72 bits wide. Port A and Port B share the same clock signal. Within a single cycle of the external clock, the Port A operation always completes before the Port B operation. Each port can independently perform either one read or one write operation per clock cycle. When both ports perform a write operation in the same clock cycle with the same address (i.e., address collision), the Port B write takes effect because the write on Port A is overwritten. When Port A performs a read while Port B performs a write with the same address, Port A gets the old data in the memory array and then the new data at Port B is written into the memory array. When Port A performs a write operation while Port B performs a read operation, the new data at Port A is written to the memory array and the new data is also read out on Port B. For each port, when executing a write operation, the read output for that port is unchanged, holding the previous value.

The content of each UltraRAM memory array is protected by a Hamming code single-error correction double-error detection (SECDED) ECC code on each of the two ports, ensuring data integrity. With ECC enabled, UltraRAM offers 64 bits wide protected data words. The UltraRAM SECDED ECC code is designed to be compatible with that of the block RAM. This can be used to enable end-to-end data-protection schemes where the data encoded at the input of one memory can remain protected across multiple pipeline stages, types of memories, and/or logic before being decoded at the output of a downstream memory.

UltraRAM contains up to four pipeline stages for each of the two ports and can be configured for one to four clock cycles of latency. When creating large RAM arrays from cascaded UltraRAMs, latency is a function of the number of UltraRAMs used—i.e., the size of the array and the target operating frequency.

Power Reduction

UltraRAM offers various built-in features to maximize power efficiency, often without user intervention. This includes the ability to:

- Power down UltraRAMs that are not used in the design
- Automatically clock-gate unused pipeline registers
- Put each UltraRAM into sleep mode when unused for extended periods of time

UltraRAMs can be manually or automatically be put into sleep mode. The user has access to the SLEEP port of the UltraRAM. When SLEEP is asserted, the UltraRAM starts entering sleep mode in the next clock cycle. In sleep mode, the peripheral logic around the UltraRAM SRAM memory is powered down but the SRAM remains powered and retains its contents.

Every UltraRAM contains circuitry that can look ahead to identify whether the block will be used for the next N cycles and, if not, the block can be put into sleep mode when the unused period is sufficient. This automatic sleep mode uses the sleep port, with its built-in control logic, to look ahead into upcoming address and enable signal states. The AUTO_SLEEP_LATENCY attribute determines how far in advance of other inputs the address and enable signals need to arrive. This information is used to determine if there is sufficient time for individual UltraRAMs to be put to sleep and wake up before their next activity.

Using UltraRAM in a Design

The UltraScale+ portfolio uses Xilinx's Vivado® Design Suite. The design tools provide a number of code templates that help the user to successfully target the available resources in a device. There are three methods to use UltraRAM in an RTL design: writing code to infer a memory; instantiating the device primitive; and using the recently added Xilinx Parameterized Macros (XPM). Most users are familiar with inference and instantiation. The Vivado Design Suite includes inference templates under:

Templates → VHDL/Verilog → Synthesis Constructs → Coding Examples → RAM

In Vivado Design Suite v2016.1. the user must specify the **ramstyle=”ultra”** to explicitly instruct Vivado synthesis to use UltraRAM.

Users who need to have ultimate control over how UltraRAM blocks are connected can use the device primitives at:

Templates > **VHDL/Verilog** > **Device Primitive Instantiation** > **Kintex/Virtex UltraScale+** > **BLOCKRAM** > **URAM**

While providing the user with the tightest control over how individual components are connected, this method can quickly become laborious if using many components with many ports and signals.

The third way of using UltraRAM in RTL designs is through XPM:

Templates → **VHDL/Verilog** → **Xilinx Paramaterized Macros (XPM)** → **Memory (XPM_MEMORY)** → **RAM**

XPM is a new tool for creating RAM and ROM structures according to user-specified requirements. Within the XPM code, the user specifies a number of generics including memory size, clocking mode, ECC mode, etc. These requirements are then converted by Vivado synthesis into the appropriate size and style of memory array.

In Vivado Design Suite v2016.1, the user must specify the value UltraRAM on the MEMORY_PRIMITIVE generic to explicitly instruct Vivado synthesis to use UltraRAM.

Creating Memory Arrays

The UltraRAM architecture is extremely scalable, allowing for a large number of UltraRAM blocks to be efficiently connected together to form deep logical memories with minimal to no fabric resources and a relatively low access latency. Each UltraRAM contains all the logic necessary to fully cascade, and optionally pipeline, the entire input as well as the output interfaces across adjacent UltraRAM blocks within the same column. Moreover, each UltraRAM contains additional logic and control inputs, per port, to independently determine whether a passing read or write transaction in an array of UltraRAM blocks is meant for the local array, or to be ignored. This form of distributed decision logic enables flexible and low-overhead scalability..

Whenever a specified memory array in a design requires more than one UltraRAM block, a decision must be made regarding how to connect the multiple UltraRAMs together. The two options are either to use all the UltraRAMs within a column, or to use the UltraRAMs in a multi-column array. If building an UltraRAM array within a column, all the necessary routing to cascade the UltraRAMs is contained within that column, enabling the tools to take advantage of the available dedicated resources. See Figure 1.

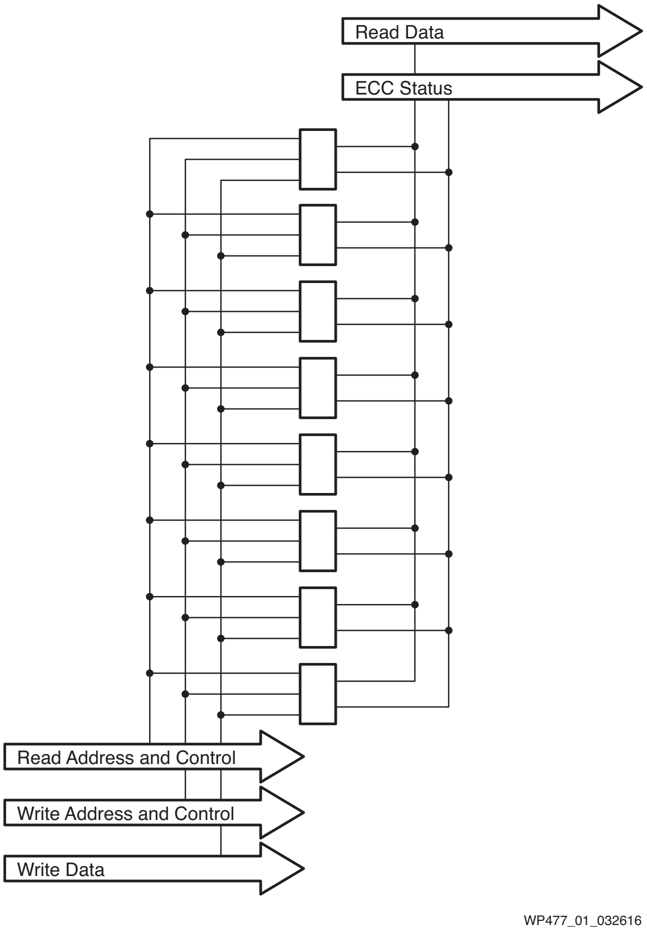

Figure 1: **UltraRAMs Cascaded within a Column**

If the design requirements determine that a multi-column array is a suitable implementation—e.g., if the required array is larger than can fit in a single column—then a multi-column array is created (see Figure 2). In this situation, it is necessary to use some fabric resources to connect the multiple columns together. The maximum achievable frequency, as well as the access latency in multi-column arrays, is a function of array size, number of columns used, and other design factors such as how many fabric resources between the UltraRAM columns are used by other parts of the design.

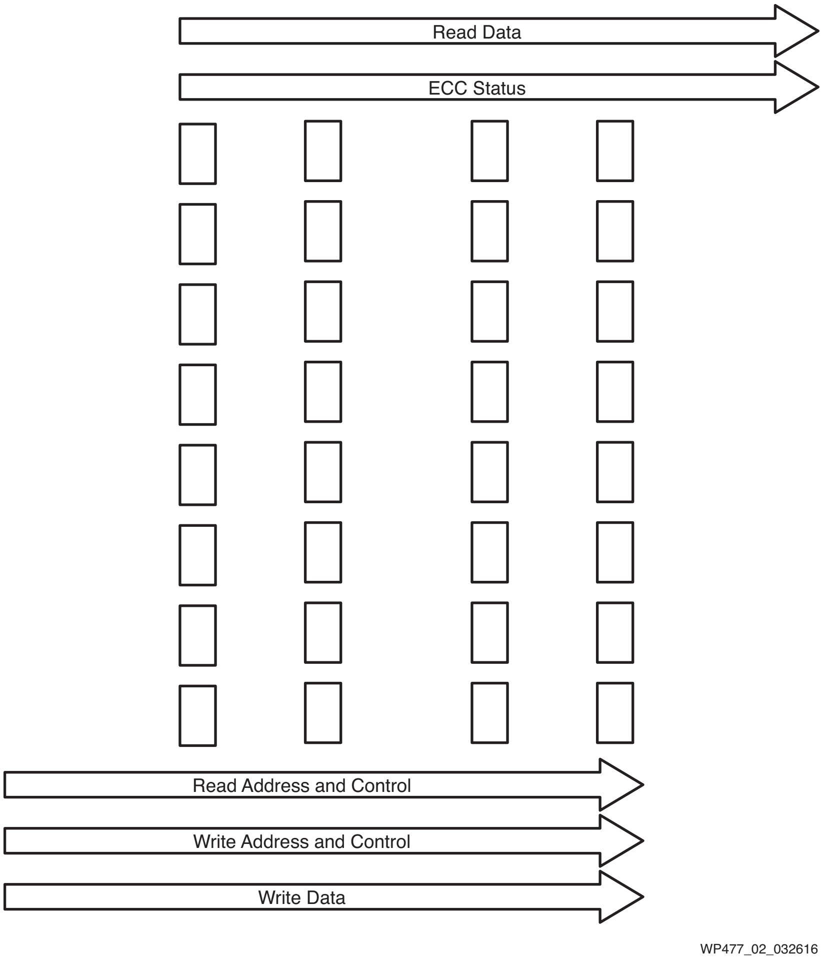

Figure 2: **Multi-Column UltraRAM Array**

When creating an UltraRAM array, the address, control, and write data signals are provided to the bottom left of the array and data is read from the top right of the array. This ensures that, by appropriately pipelining an array of fixed size, the latency to read any data from any address is always the same.

Wide data or control path designs can require more bus width than a 64-bit ECC protected or 72-bit non-ECC protected word width that each UltraRAM can natively offer. In such cases, multiple identical instances of the UltraRAM block or the cascade structures described above, can be used in parallel to build very efficient wide word data storage structures.

Registers and Pipelining

Every UltraRAM contains several input and output registers, shown in Figure 3, some of which are optional depending on the configuration and desired functionality.

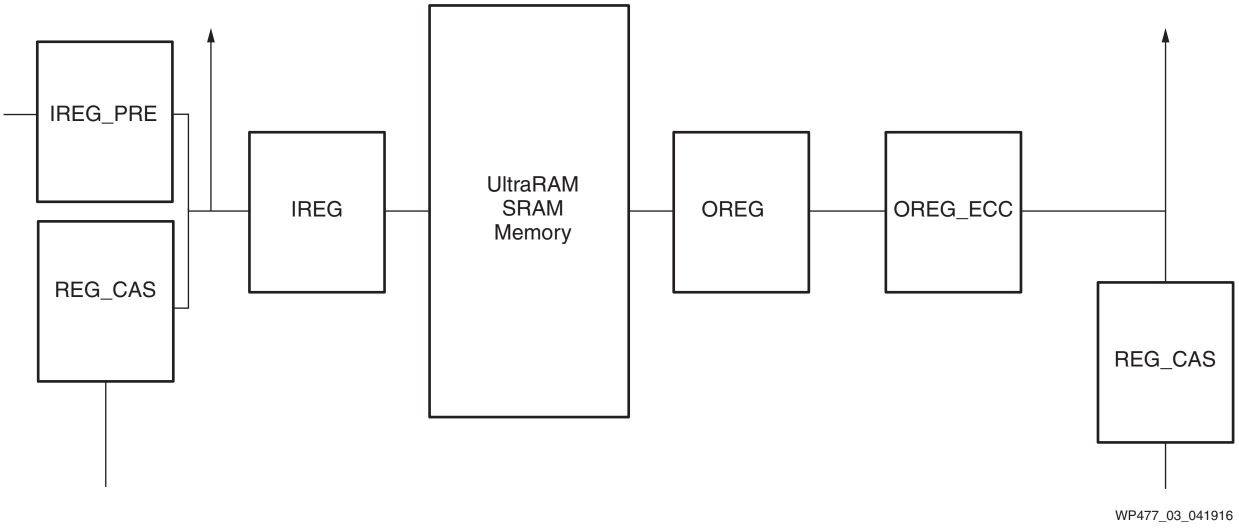

Figure 3: **UltraRAM Registers**

The IREG, OREG, and OREG_ECC registers are activated based on the memory functionality desired by the user. In addition, there are registers for use when cascading multiple UltraRAMs together: IREG_PRE and REG_CAS.

When connecting multiple UltraRAMs together, certain rules need to be followed to maintain the maximum frequency of operation:

- The first instance in a single column array or the first instance in each column of a multi-column array must use the IREG_PRE register.
- If required in the design, the OREG and/or OREG_ECC registers should be enabled consistently for all UltraRAMs in the array.
- The REG_CAS register must be activated every four UltraRAM blocks.
- The last instance in a single column array or the last instance in each column of a multi-column array must use the REG_CAS register.
- If the array spans vertically across multiple clock regions, additional pipeline stages are required on either side of the clock region boundary. There are sixteen UltraRAMs per clock region per column.

For more information and detailed diagrams of the UltraRAM block, refer to chapter 2 of UG573, *UltraScale Architecture Memory Resources User Guide.*

Migrating to UltraRAM

It is important for users to know the quantity of the different memory resources available in their chosen target device. In some circumstances, a user can initially target a device that only has block RAM but the design will ultimately target a device that contains UltraRAM. In this situation, users need to understand the features of UltraRAM and write their code to use the features in both block RAM and UltraRAM. This way, the tools can use UltraRAM when it is available in the targeted device and the user will not have to redesign when changing to a different device.

Table 1 illustrates how much UltraRAM is available in the different UltraScale+ families. In Virtex UltraScale+ devices, it is possible to connect all the UltraRAMs together to create a multi-column array of up to 360Mb, although more common use models of multiple instances in the 10Mb to 144Mb range are also possible.

Table 1: **UltraRAM Resources in UltraScale+ Portfolio**

Family	Kintex UltraScale+	Virtex UltraScale+	Zynq UltraScale+
UltraRAMs	0–128	320–1,280	0–120
UltraRAM (Mb)	0–36	90–360	0–36
Column Height (Mb)	0–36	18.0 or 22.5	0–36

All UltraScale+ devices also contain block RAM and distributed RAM for smaller memories, enabling over 500Mb of total on-chip memory in the VU13P.

DS890, *UltraScale Architecture and Product Overview*, presents further details on device resources.

If UltraRAM is available in the chosen device, it should be used for memories of 144Kb and larger.

Replacing External Memory Components

With up to 360Mb of UltraRAM that can be configured in an almost infinite number of permutations, there are many potential use models. One popular UltraRAM use model is to incorporate memory into the FPGA or MPSoC that would otherwise have been implemented in an external memory such as a QDR SRAM. The advantage of having the memory on-chip is that it can be the exact size required, as opposed to the size the memory vendor supplies, and can be logically very close to where the data is buffered or stored in the design. Additionally, to interface to an external memory, the data must travel through a memory interface or controller and through I/O pins on both the FPGA/MPSoC, with the external memory component adding significant power consumption to the system. By bringing the large storage capacity on-chip in the form of UltraRAM, the power consumed by the I/O in the FPGA/MPSoC and the I/O in the external memory is eliminated.

Input Multicast

Each UltraRAM port can locally determine whether a transaction in cascade mode is bound for its local memory array. This ability can be used to enable dynamic, transaction-level write unicast, multicast, and broadcast capable memory structures. Such structures can form the basis for efficient output buffered memory switches.

One UltraRAM memory port is dedicated to writes while the other is dedicated to reads. The write ports across all UltraRAM blocks are configured for one contiguous cascade using a unique one-hot encoded SELF_ADDR and a matching one-cold encoded SELF_MASK value per block. Each write transaction can be sent to a single, multiple, or all UltraRAM blocks, by setting one or more corresponding global / upper address bits matching the logical-OR of the SELF_ADDR attribute of all target blocks. The read ports of all UltraRAM blocks are non-cascaded, effectively allowing a single write transaction to potentially propagate to multiple consumers.

Each UltraRAM block represents the buffering requirement for one consumer. For greater storage capacity, each UltraRAM block can instead be a cascade of *N* UltraRAM blocks per consumer. Assuming *C* consumers read in parallel from *C* logical memories that are *N* UltraRAM blocks deep each, the write port requires a contiguous cascade across all *C* * *N* UltraRAM blocks while the read port requires *C* independent cascades, each *N* UltraRAMs deep.

Figure 4 shows one such configuration with four outputs / consumers, each requiring a single UltraRAM block of buffering and two sample multicast transactions bound for UltraRAM blocks 0 and 1 as well as 1 and 3, respectively.

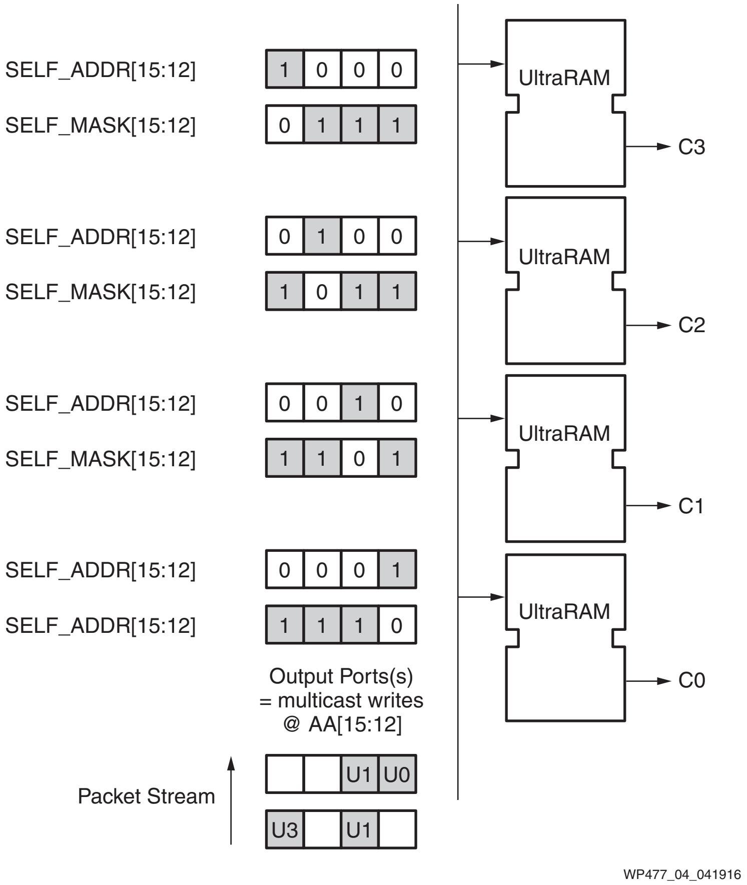

Figure 4: **Input Multicast**

Conclusion

Every design needs and uses memory in some shape or another. The addition of large, flexible UltraRAM blocks to the UltraScale+ portfolio provides users with greater on-chip storage than any preceding or competing devices. The ability to connect blocks together to form memory arrays of different sizes enables users to create the ideal memory solution for their design, precisely where it is needed eliminating the power and board complexity of using external SRAM components.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
06/14/2016	v1.0	Initial Xilinx release.

Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos.

Automotive Applications Disclaimer

XILINX PRODUCTS ARE NOT DESIGNED OR INTENDED TO BE FAIL-SAFE, OR FOR USE IN ANY APPLICATION REQUIRING FAIL-SAFE PERFORMANCE, SUCH AS APPLICATIONS RELATED TO: (I) THE DEPLOYMENT OF AIRBAGS, (II) CONTROL OF A VEHICLE, UNLESS THERE IS A FAIL-SAFE OR REDUNDANCY FEATURE (WHICH DOES NOT INCLUDE USE OF SOFTWARE IN THE XILINX DEVICE TO IMPLEMENT THE REDUNDANCY) AND A WARNING SIGNAL UPON FAILURE TO THE OPERATOR, OR (III) USES THAT COULD LEAD TO DEATH OR PERSONAL INJURY. CUSTOMER ASSUMES THE SOLE RISK AND LIABILITY OF ANY USE OF XILINX PRODUCTS IN SUCH APPLICATIONS.