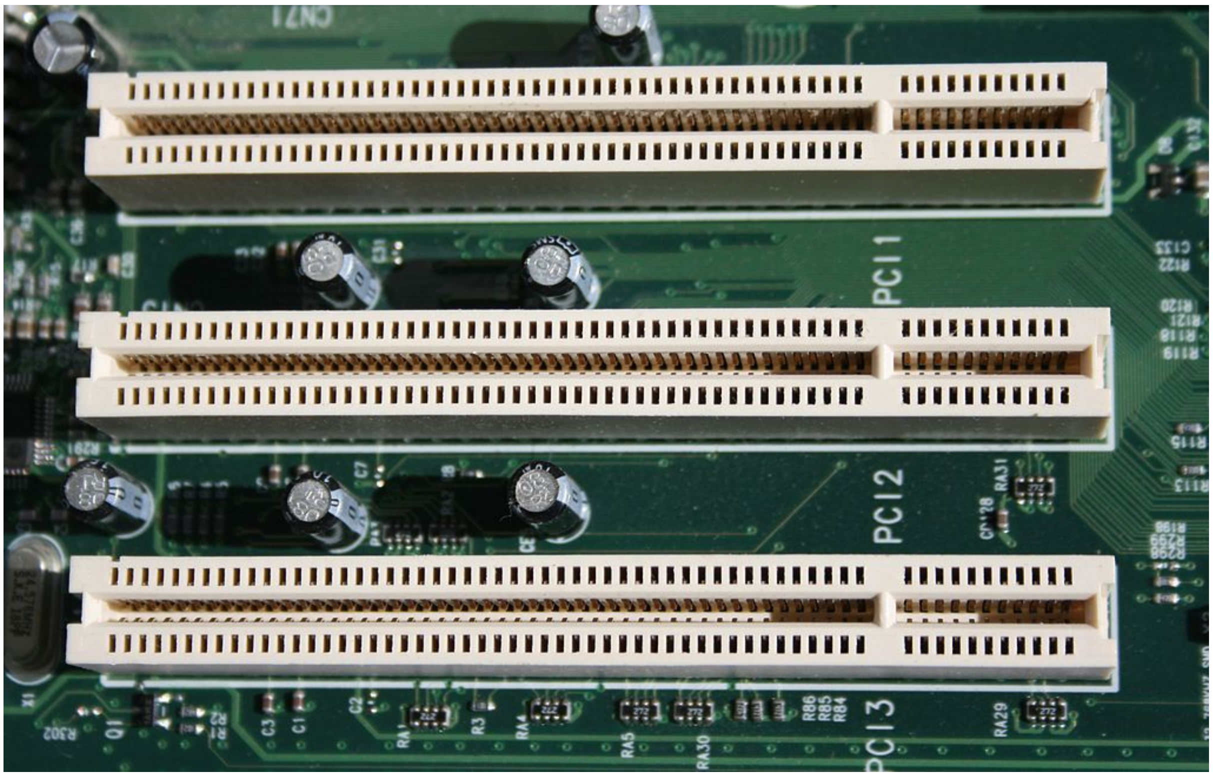
PCI

project report

1.overview & history



Conventional PCI, often shortened to PCI, is a local computer bus for attaching hardware devices in a computer. PCI is the acronym for Peripheral Component Interconnect and is part of the PCI Local Bus standard. The PCI bus supports the functions found on a processor bus but in a standardized format that is independent of any particular processor's native bus. Devices connected to the PCI bus appear to a bus master to be connected directly to its own bus and are assigned addresses in the processor's address spac. It is a parallel bus, synchronous to a single bus clock.

Attached devices can take either the form of an integrated circuit fitted onto the motherboard itself (called a *planar device* in the PCI specification) or an expansion card that fits into a slot. The PCI Local Bus was first implemented in IBM PC compatibles, where it displaced the combination of several slow ISA slots and one fast VESA Local Bus slot as the bus configuration. It has subsequently been adopted for other computer types. Typical PCI cards used in PCs include: network cards, sound cards, modems, extra ports such as USB or serial, TV tuner cards and disk controllers. PCI video cards replaced ISA and VESA cards until growing bandwidth requirements outgrew the capabilities of PCI. The preferred interface for video cards then became AGP, itself a superset of conventional PCI, before giving way to PCI Express.

1.1 PCI bus transaction

PCI bus traffic consists of a series of PCI bus transactions. Each transaction consists of an *address phase* followed by one or more *data phases*. The direction of the data phases may be from initiator to target (write transaction) or vice versa (read transaction), but all of the data phases must be in the same direction. Either party may pause or halt the data phases at any point. (One common example is a low-performance PCI device that does not support burst transactions, and always halts a transaction after the first data phase.)

Any PCI device may initiate a transaction. First, it must request permission from a PCI bus arbiter on the motherboard. The arbiter grants permission to one of the requesting devices. The initiator begins the address phase by broadcasting a 32-bit address plus a 4-bit command code, then waits for a target to respond. All other devices examine this address and one of them responds a few cycles later.

64-bit addressing is done using a two-stage address phase. The initiator broadcasts the low 32 address bits, accompanied by a special "dual address cycle" command code. Devices which do not support 64-bit addressing can simply not respond to that command code. The next cycle, the initiator transmits the high 32 address bits, plus the real command code. The transaction operates identically from that point on. To ensure compatibility with 32-bit PCI devices, it is forbidden to use a dual address cycle if not necessary, i.e. if the high-order address bits are all zero.

While the PCI bus transfers 32 bits per data phase, the initiator transmits 4 active-low byte enable signals indicating which 8-bit bytes are to be considered significant. In particular, a write must affect only the enabled bytes in the target PCI device. They are of little importance for memory reads, but I/O reads might have side effects. The PCI standard explicitly allows a data phase with no bytes enabled.

2.Modules

we have two modules in our design

2.1 Device

Device has two modes it can works as initiator (master) and target .
if it works as initiator the device sends a signal (REQUEST) to arbiter to give the device the grant then we enter the operation of the transaction , we have two option (write & read)

2.2 arbiter

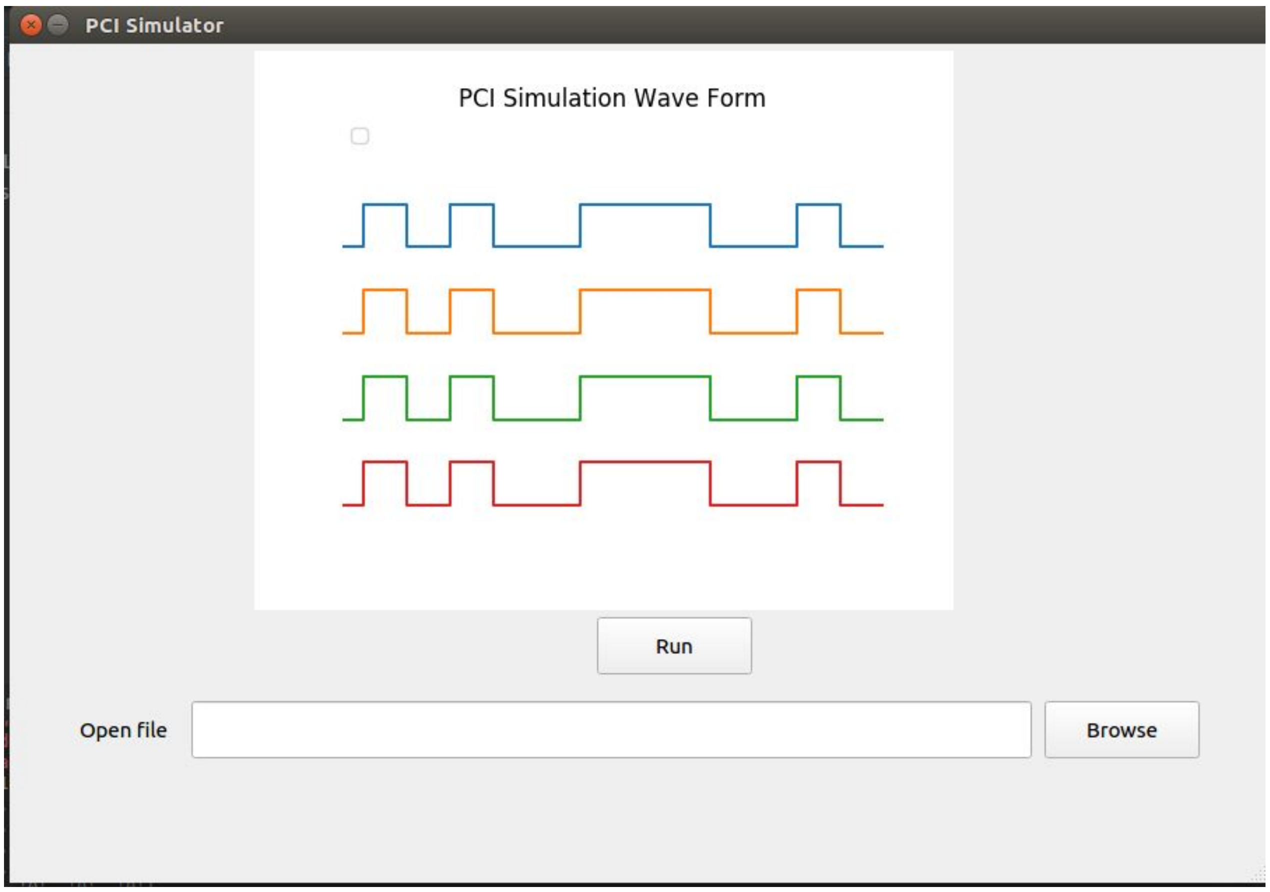
Any device on a PCI bus that is capable of acting as a bus master may initiate a transaction with any other device. To ensure that only one transaction is initiated at a time, each master must first wait for a bus grant signal, GNT#, from an arbiter located on the motherboard. Each device has a separate request line REQ# that requests the bus, but the arbiter may "park" the bus grant signal at any device if there are no current requests.

The arbiter may remove GNT# at any time. A device which loses GNT# may complete its current transaction, but may not start one (by asserting FRAME#) unless it observes GNT# asserted the cycle before it begins.

The arbiter may also provide GNT# at any time, including during another master's transaction. During a transaction, either FRAME# or IRDY# or both are asserted; when both are deasserted, the bus is idle. A device may initiate a transaction at any time that GNT# is asserted and the bus is idle.

PCI GUI simulator

We designed a GUI PCI to simulate the bus lines on a User friendly wave form data, we used PyQt5, a software binding from QT C++ framework, the App is a cross platform app which works on any Windows, linux or Mac.


PCI Simulator
PCI Simulation Wave Form
Run
Open file
Browse