Moving from the ZOZ to the WIZ

Back to Index

From ZOZ to WIZ

Chapter 1: Singular and Generic

NEXT

ZOZ - Singular and Generic - page 1

Every ZOZ Tap has a serial number, assigned at manufacture. It is a unique-in-all-the-galaxy serial number, identifying that particular Tap (and its associated object or device) and no other. This type of serial number is what I call "Singular".

For example, if I want to turn on the light on my desk, I supply a specific singular serial number. I want *that* light and no other. An identical light a few centimeters away is not the one I want.

But let's consider a multiplier. If I need to multiply some numbers and there are many identically-manufactured multipliers of the specific type I need, it would not matter which one I use. These multipliers *can* be substituted for one another.

We shall give such a set of identically-manufactured identically-functioning devices like these multipliers a second serial number, one that identifies the entire set with a single number.

This second serial number I call a "Generic" serial number. When we manufacture the Taps for these types of devices, we give them TWO serial numbers: one, their unique or Singular serial number, and two, a Generic serial number, common to all the devices with the same identically-manufactured function.

NEXT

As a comparative example, take the UPC or "Universal Product Code" labels found on most current products. For example, all 10 oz bags of "Fritos Original Corn Chips" have labels with the number 028400039246 on them. If these were ZOZ products, that code would be their generic serial number. All bags or boxes with this same code are identical (or at least identical enough for practical purposes). So if I wanted a 10 oz bag of "Fritos Original Corn Chips", I could specify that code, and any such labelled bag would do.

And of course, if these were ZOZ products, each bag would also have a unique or Singular ID number as well. Using that serial number, I could specify one specific bag and no other. This is very impractical to do on bags of corn chips, but easy to do with electronic devices.

Almost all arithmetic devices (like adders, subtractors, dividers, square rooters, doublers, etc) have generic serial numbers. There can be zillions of each of them, spread throughout the galaxy. And when one is needed, the ZOZ can choose the closest or most convenient one it finds.

And so while there may actually be a "doubler" object on Pluto, there may also be many more identical doublers all over the galaxy. Each has a different singular serial number, but all have the same generic serial number. If I want to route a datum to the doubler, I would specify its generic serial number, and let the ZOZ find the closest one.

NEXT

ZOZ - Singular and Generic - page 3

Memory devices are interesting. They have THREE serial numbers!

Two are assigned to the hardware when manufactured, a singular one and a generic one.

But there will be a third, generated dynamically by the ZOZ.

When a memory device is empty, having no data written into it yet, it is generic. If I have some data to be written, I can let the ZOZ choose any suitable generic and empty memory device of the type I need.

But the moment I write some data into it, it becomes unique. That particular data is stored only in that particular memory device, at least for the moment. So the moment data is written to a memory device, the ZOZ creates a new serial number and assigns it, not to the device but TO THE DATA ITSELF!

Now this is a new concept: the data itself has a serial number. *Data* is not an object or device. Yet it has its own serial number!

Thus, a memory device has three serial numbers: (1) its own Singular one; (2) its own Generic one; and (3) the serial number of the data currently stored in it.

NEXT

And it gets even more interesting: I can copy data from one memory device to another. A set of data can be duplicated into many memory devices at once, and so long as it is not further modified, I can read it back from any of them. Thus, the serial number created for data is actually generic -- any copy of it will do.

A sequence of instructions stored into a sequencer, like the "squarer" sequence or the "CtoF" sequence previously described, is likewise generic. What we call "the squarer" is not a device, it is a sequence of numbers which can be moved to any suitable sequencer.

So when we route some data to, for example, the "squarer", we are looking not for a particular sequencer device, sitting out in the data stream waiting to be triggered, but for any sequencer device which currently has a copy of the "squarer" sequence stored in it. The "squarer" is not a device at all: it is the instruction sequence itself.

The ZOZ can even find a nearby empty sequencer and copy the required sequence of instructions into it, thus creating generic copies of functions whenever and wherever they are needed.

Back to Index

From ZOZ to WIZ

Chapter 2: The ZOZ generic chip

NEXT

ZOZ - The ZOZ generic chip - page 1

A ZOZ toaster object has to be the size of a toaster, or the toast wouldn't fit into it. And a ZOZ thermometer object has to be at a specific location, or it would be measuring some other location's temperature.

But most computational devices, like adders and multipliers, and memory devices and sequencer devices, can be any size and shape and at any location. And being generic, we can have any number of them.

Current manufacturing techniques can make most of these generic computational devices very inexpensively and at a very tiny scale, just *billionths* of an inch. We can deposit millions of them onto a single "chip" of silicon less than a millimeter across.

Thus it is possible, even easy, to collect together into one place, millions of various computational and generic ZOZ devices, not spread out all over the galaxy, but compressed into the space of less than a single cubic millimeter. Such a collection would be like a miniature ZOZ "galaxy" unto itself.

NEXT

ZOZ - The ZOZ generic chip - page 2

On such a chip we can connect the numerous ZOZ devices with a microscopic wire (called a "bus") to create a tiny ZOZ data stream running through the inside of the chip. Because of its small size and direct connections, data moving from one device to another along this wire can move extremely rapidly.

This tiny ZOZ data stream inside this tiny chip then connects to the rest of the ZOZ data stream outside the chip, through a wire extending out from the chip, or via a tiny radio inside the chip that communicates with other radios outside the chip or with other similar chips in the ZOZ data stream.

Also inside the chip can be a very tiny battery, or some form of power generator, or a power collector which absorbs (or "harvests") energy directly from the environment, which might allow the chip to run without any external power source.

And, being very inexpensive to manufacture, we can create zillions of such chips and locate them throughout the galaxy. We can place one or more of them inside almost every larger ZOZ object (like thermometers and toasters), enabling algorithms associated with those objects to run very efficiently.

NEXT

ZOZ - The ZOZ generic chip - page 3

When a sequencer inside such a chip streams out an instruction with two generic serial numbers like "adder => multiplier", the ZOZ first looks for devices with those generic serial numbers right there inside that chip. If found, the instruction can use those two devices and move that datum very rapidly between them, as it would be going only a very short distance along a very high-speed wire.

It would not need to use the radio or impinge upon the rest of the ZOZ outside the chip. That is, all the action of this instruction would occur completely within the boundaries of the chip.

Sequences in which most if not all of their instructions reference computationally generic devices are extremely common; and if we can put literally millions of such generic devices, of a wide variety of types, into a single chip, the probability may be extremely high that we will see most if not all of our activity operating within the boundaries of single chips.

An occasional instruction which references a sensor or motor or other serial number which is not inside the chip would travel, via wire or radio, onto the larger ZOZ data stream outside the chip. But the great majority of instructions might execute and communicate completely within the boundary of the chip, thereby running very fast and creating almost no traffic on the greater ZOZ data stream outside the chip.

NEXT

ZOZ - The ZOZ generic chip - page 4

I call this chip a "WIZ" chip.

A "WIZ" chip is tiny chip, perhaps a fraction of a millimeter across, which collects together into one location a number of tiny ZOZ devices, particularly generic devices like arithmetic, memory and sequencer devices.

A WIZ is not different from the ZOZ -- it is merely a big piece of the ZOZ, shrunk into a tiny space.

It would generally be manufactured out of silicon-based electronic circuits, and would often also contain a radio and a power source.

WIZ chips can vary as to how big they are, how many devices they contain, and which specific devices they contain.

In the following sections, I will demonstrate that such a WIZ chip will function quite favorably as what Intel and other companies refer to as a "general-purpose microprocessor". Indeed I will show that a WIZ can significantly out-perform these microprocessors with regard to almost every category of measurement, such as speed, energy usage, manufacturability, reliability, design ease, programming ease, and more.

Back to Index

From ZOZ to WIZ

Chapter 3: Many WIZes on a WIZ chip

NEXT

Many WIZes on a WIZ chip - page 1

So a WIZ chip contains a number of nano-scale ZOZ devices collected together onto a tiny piece of silicon. The ZOZ data stream inside a WIZ chip is implemented by a high speed bus, which connects to the ZOZ data stream outside the chip using a wire or radio or similar device.

I shall now add a bit more detail.

WIZ chips are divided into a large number of sections. Each section contains no more than 256 devices, at least one of which is a memory sequencer. Each section has its own local bus connecting its 256 or fewer devices. A higher-level bus then runs through all the sections of the chip, connecting them all to each other and to the greater ZOZ data stream outside the chip.

Each section may vary as to how many (up to 256) and what kinds of devices are in it, but every section must have at least one sequencer.

All of this is built into the WIZ chip when it is manufactured. Each manufacturer may vary the design as to how many sections, how many devices in each section, and what specific devices are in each section.

NEXT

Many WIZes on a WIZ chip - page 2

When a section's sequencer streams out an instruction, it may be that both the source and destination devices for that instruction can be found not just within the same chip, but within the same section of the chip. In this case, only the one local bus in that section will be involved in the routing of the datum between them.

If most or all of the instructions in a sequence can be satisfied by devices on its local bus, that sequence may run in virtually complete isolation from other sequences running (simultaneously) in other sections of the same WIZ chip.

With such sectioning, we can run many sequences simultaneously. Each section, with its own sequencer controlling its own bus with its own collection of ZOZ devices, can each be running its sequence independently of the rest of the chip and the rest of the ZOZ outside the chip.

Each section, with its sequencer and bus and up to 256 devices, is called a "WIZ". The entire chip is called a "WIZ chip". Thus, a "WIZ chip" contains many "WIZ"s.

A WIZ chip is then a massively parallel set of WIZs.

From ZOZ to WIZ

Chapter 4: ZOZ Instruction Format

NEXT

ZOZ instruction format - page 1

A ZOZ datum, serial number, or absolute number, can be very large -- there is no limit to the number of digits in a single number.

We represent them with a variable length string of 16-bit words, prefaced by two words of metadata: (1) the count, "N", of the number of words in the datum, and (2) a 16-bit CRC checksum of the entire string of words.

In other words, any ZOZ datum or number is stored in N+2 words, like this:

```
   N , CRC , datum word 1 , datum word 2 , ... , datum word N
```

Since N is stored in a 16-bit word, the longest datum possible in this format is just under 2^16 or about 64K words.

NEXT

ZOZ instruction format - page 2

A ZOZ instruction consists of a pair of such numbers, a source and a destination. It is formatted with a header containing metadata about the instruction, followed by two variable length numbers in the format just given, all concatenated into a single string of words. That is:

```
   H,  CRC0,
   N1, CRC1, datum1 word 1, datum1 word 2, ..., datum1 word N1,
   N2, CRC2, datum2 word 1, datum2 word 2, ..., datum2 word N2.
```

All items here (between commas) are 16-bit words. H is the header word (detailed on the next page), CRC0 is a CRC (checksum) of the entire instruction including the header; N1 and N2 are the lengths (in number of words) of the first and second numbers, CRC1 and CRC2 are their respective CRC checksums, and datum1 and datum2 are the two numbers, split into multiple words of length N1 and N2 respectively.

These two numbers are of course the source serial number (or absolute number) and the destination serial number.

Thus the total length of a ZOZ instruction is N1+N2+6 words; N1+N2 is the length of the numbers themselves, and H, CRC0, N1, CRC1, N2, and CRC2 are 6 extra words of metadata.

This format is used in all contexts, including when instructions are stored in a sequencer's memory or any memory or when transmitted along the ZOZ data stream. It is a self-contained "packet". No other type of packet or protocol is ever used within the ZOZ.

NEXT

ZOZ instruction format - page 3

The instruction header ("H") is a single 16-bit word. Its high bit is the "absolute number" bit. If 1, it specifies that the instruction's source number is an absolute number rather than a serial number (for example, the "50" in an instruction like "(50) => fan".)

The remaining 15 bits of the header word are reserved for later definition.

And that is all there is to say here. Its a very simple format, basically just two numbers and a small amount of overhead for length and checksum metadata.

Back to Index

From ZOZ to WIZ

Chapter 5: WIZ instruction format

NEXT

WIZ instruction format - page 1

All taps have a unique-in-all-the-galaxy serial number. These can be very long numbers of many digits each, and are stored in a variable length format of multiple 16-bit words. Thus a ZOZ instruction, containing two variable length numbers plus some metadata, can be very long.

But because a WIZ has a limit of 256 taps on its local bus, each tap also has an 8-bit "local ID", a number from 0 to 255, which also uniquely identifies each tap. In the context of a particular WIZ, the local ID of any tap can be used interchangeably with its serial number.

When the ZOZ stores instructions in the memory of a WIZ sequencer, it looks for instructions which have source and destination serial numbers which both exist on that WIZ's local bus. If so, the ZOZ doesn't store the full instruction, but instead stores a much shorter format instruction which uses only the 8-bit local IDs of the source and destination taps.

This is done in the context of each WIZ separately. The same sequence, loaded into a different WIZ with different devices on its local bus, would produce a different result.

These much shorter instructions are called "WIZ instructions" rather than "ZOZ instructions".

NEXT

WIZ instruction format - page 2

WIZ instructions are all single 16-bit words, with the source's local ID in the left 8 bits and the destination's local ID in the right 8 bits.

And that is all there is to it! We drop the metadata, checksums, etc, and just have the pure instruction. This very simple format allows for very fast hardware execution.

Of course, not all ZOZ instructions can be converted to WIZ instructions. This would be because one or both of the ZOZ instruction's serial numbers refer to devices which are not on this WIZ and do not have an equivalent local device ID. (For example, a sensor far away). In this case, a special WIZ instruction is inserted, followed by the full ZOZ instruction.

When the WIZ executes a sequence, all WIZ instructions execute directly and at very high speeds, and all ZOZ instructions are passed back to the ZOZ for execution elsewhere.

Of course, and especially in the case of arithmetic and computational instructions, we expect that a very large proportion of a sequence's instructions, often *all* of them, will be reducible to WIZ instructions and therefore run locally without ever going back to the ZOZ.

Back to Index

From ZOZ to WIZ

Chapter 6: Outro

NEXT

WIZ - Outro - page 1

Thus, as our title promises, we have gone from ZOZ to WIZ.

A WIZ is just a piece of the ZOZ, a group of various ZOZ devices collected together into a single location, a tiny chip of silicon, executing instructions which copy data almost exclusively amongst themselves.

It is a kind of almost-stand-alone microprocessor. The 16-bit WIZ instruction words are its operation codes, and the set of all combinations of local devices are its instruction set, unique to and optimized for each WIZ independently.

And thus we have taken a galactic scale ZOZ and condensed a subset of it into a volume on the scale of a cubic millimeter, allowing it to execute ZOZ instructions as 16-bit WIZ instructions with almost no overhead on the smallest and fastest hardware current Earth Technology can produce.

Back to Index