P2 Introduction For Forth Developers

P2 is an inexpensive ManyCore Processor

The Parallax P2 Propeller is an inexpensive ($15) manycore processor with 8 cores (cogs) surrounding a central hub. Each cog has its own memory, and shared access to the central hub's memory, and other resources. The P2 includes 64 I/O pins, , making it excellent for IoT applications.

Market Niches

The P2 excels in several market niches.

1. Hard real time control. Where you do not want the complexity of multiplexing multiple I/o's on a single core.
2. High Frequency, the competing chips max out at 80Mhz for data I/O. The P2 can do data I/O at up to 1GigaByte per core, even with analog output.
3. Complex I/O. Where the smart pin functionality is needed.
4. ManyCore. The 8 300Mhz cores can all work on the same data cooperatively.
5. Cordic calculations of Sine and Cosine.

Competitors

There are a number of competitors, but none of them are nearly as competent. More importantly a lot of the smart people who understand the limits of the other chips, move over to the Prallax Forums.

1. ESP32 is a very cheap chip, but it is only a single processor. Software wise, it is very hard to do different things in real time on a single processor. So much easier to dedicate one processor to each real time process. And the ESP32 pins are on a bus, limited to 20Mhz divided by the number of pins.
2. XMOS. A very interesting chip, but they split the memory into tiles, but the band width between tiles is only 1 bit per clock cycle. And they can time slice a core between 8 processes, which is good, but then each one only gets 1/8th of the processor.

Chip Architecture

The P2 has 8 cogs sharing access to a central hub.

The central hub includes Hub Ram, a Cordic math solver, 16Kb of ROM, 64 smart I/O Pinsl

Each cog includes a 32 bit core, 512 32 bit (long) registers, 512 longs of Look Up table RAM. (LUTRAM), a streamer for data and a FIFO for faster hub RAM access.

There are also ways for the cogs to communicate.

Hub Ram

Hub RAM is divided into n slices. One slice for each Cog/Core. Meaning 8 slices on the P2. The cogs all access HubRAM simultaneously in a round robin fashion, each accessing a different slice. I like to think of them a bit like a hard drive with 8 heads. As the propeller spins the cogs get round robin access to each memory block. HubRam slices are accessed using 20 bit addresses. The lowest 2 bits specify the byte within a long. The next lowest 3 bits specify the slice on the P2. This way consecutive words (sets of 4 byte-level addresses) are in consecutive slices. Making it easy to read a large region of memory on consecutive clocks one word at a time. This is used by the FIFO buffer discussed next.

FIFO

There are two ways to access HubRAM. It can be accessed directly by the cogs/cores, but that involves an average delay of (n-1)/2 clock cycles (3.5 on the P2). For improved performance, a region of memory can be read or written all at once using the FIFO buffers. Each cog has its own FIFO buffer.The FIFO buffer reads or writes a word on every clock from consecutive slices. While it still takes 3.5 clock cycles to get the first word from the FIFO, after that it never runs out. Unless you want it to. Sadly jumping to a new address, costs the 3.5 clock delay, so FIFO also allows for instructions skipping.

STREAMER

Much like Fifo speeds up access to HubRam, the streamers speed up access to the pins. Streamers can read or write data from LUTRAM to the pins, or to and from the FIFO and the Pins. This allows for very fast data I/O, and also frees up the cogs to do other things. When the boards are operating at 300 Mhz, the streamer can read or write digital or analog data every clock cycle, meaning every 3.3 nano seconds. Phenomenal. For comparison, Human speech goes up to 20 Khz.

The P2 natively supports Video, and USB, so it makes for a complete desktop computer, albeit with a lack of natively running applications.

The P2 also runs a number of programming languages: C, PASM, Python, Basic and Spin2. Each has their strengths and weaknesses.

INTER COG COMMUNIcATION

There are 4 ways for the cogs to communicate: shared hub memory, shared cog memory, 16 messaging bits, and 64 I/O bits. One cog can write to a memory address, or stream to a block of memory addresses. The next cog in the round robin cycle can read that memory on the next instruction. One cog can wait for a non-zero instruction pointer in Hub RAM. Another cog can write the instruction pointer to hub memory. There is also shared cog memory. The 8 cogs are organzied as 4 pairs. The partner cogs can allow the partner to write to their 512 words of LUT RAM. There are 16 bits for cogs to message each other. One cog can write to all 64 bits of Digital I/O pins, another cog can read it. And each cog can support up to 3 interrupts which can be invoked by the other cogs.

WiFi

There are two different options for WiFi access. The parallax supported solution is to connect an ESP32 to the P2, and use its wifi. The other option is to use the W5500 and Taqoz reloaded has the words to drive it. Then there is only a single microprocessor to program.

P2 Programming

There are multiple programming languages supported on the P2. It is important to choose the right one for the job.

Existing Applications

The easiest application to write is the one which already exists. Here is the official list, called the object exchange.

TAQOZ

IMHO opinion, the best choice of a language on the p2 is Taqoz Forth. Taqoz is a Forth dialect, written by Peter Jakacki which can run on up to 8 Parallaxx P2 cores. Taqoz Forth is small, fast and beautifully documented. Here are the links needed to get started.

By bob_g4bby

- Using Taqoz Forth: A beautiful introduction to the P2 from a Forth perspective.
- Taqoz glossary: Defines the Forth words supported by Taqoz.
- Writing Assembly in Forth: the easiest way to write and test P2 assembly is from Forth.

Peter Jakacki's introduction: teaches you Forth using the P2.
Peter's Dropbox: The official site for many of these Forth programs.

P2 Lexicon: Describes many of the terms used for the P2, and by electrical engineers using the P2.

Parallax P2 Documentation: One .of 3 official documents. This one describes the assembly language.

Spin2 cumentation: Spin2 is the programming language provided by Parallax Inc.A second part of the official documentation.

Smart Pins Documentation: The analog and digital pins support a lot of functionality, ready about it here.

Parallax Forum – Parallax and it's users provide excellent tech support here.

P2 Eval Board and accessory set.

TAQOZ RELOADED – an extended version of the language to optionally support a full fat32 file system, sound, video and much more that can be saved on flash or SD card.

HC05 Configurator - windows software for setting up Bluetooth to serial adapters

Tera term - Windows terminal program for communicating with TAQOZ

Micro Python

Micro Python is a supported language on the P2, but it does consume a lot of memory. Python has an issue of garbage collection on real time systems.

C Programming Language.

The P2 has multiple C compilers. GCC, FlexProp and …

PASM

To understand the P2, you really need to understand its assembly language. Yes, the P2 will run C programs, but the chip can do so much more. The P2 supports Byte Code interpreters, without affecting the return stacks. And it can do things in parallel. Communicate to the pins using the streamer, or with the hub using the FIFO buffer. Or skip some of the instructions fed to it by the FIFO buffer.

With C, you can force some of the code to be in Registers, or LutRam. With PASM you can move code around, code uses relative addresses to work even when moved.

Let us start by understanding the address space. It is addressed using HexaDecimal. For those who do not use hex much, here is how we count to 20.
0 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14

To distinguish hex from base 10 use $ and # respectively.
$10 = #16

Data is 32 bits, but the address space is only 20 bits wide (1 MegaBit). And the P2 only has 512Kb of Hub Ram. Those are the right (?) most bits of the long word. Represented as [20:0], Remember it is the cogs that access the address space. The first 512 32 bit long words are register space (Hex addresses $000 to $200) The next 512 x 32 bit long words are LUT RAM, $201 to $400 and from there up, one can access Hub ram on Byte boundaries. In contrast the Register and Lut Rams can only be accessed on 32 bit long word boundaries.

The last 16 words of register space have 8 dual purpose registers, and 8 special purpose registers.

The 8 cogs are quite fast when accessing Registers and LUTRAM. . They have a 5 stage pipeline. While this instruction is executing, the next instruction can be read from Register Ram. In contrast accessing HubRAM imposes a delay, it can be mitigated with the FIFO, but jumps again invoke the delay.