1. Description

The PE80 is a Z80 compliant processor soft-macro - IP block that can be implemented in digital or mixed signal ASIC designs. The Z80 and its derivatives and clones make up one of the most commonly used CPU families of all time. It was developed by the original manufacturer in 1976.
The PE80 offers higher system throughput and more efficient memory utilization than comparable second- and third-generation microprocessors.

The clock speed can be anywhere between 6 – 40 MHz, depending on technology and peripheral circuitry. VHDL synthesized gate count is in the order of 10k equivalent NAND2 gates.
The IP core suits a wide range of applications from gaming consoles to industrial automation and wireless communication as well as sensor signal conditioning.

The internal registers contain 208 bits of read/write memory that are accessible to the programmer. These registers include two sets of six general purpose registers which may be used individually as either 8-bit registers or as 16-bit register pairs. In addition, there are two sets of accumulator and flag registers.
The PE80 CPU contains a Stack Pointer, Program Counter, two index registers, a REFRESH register, and an INTERRUPT register. The CPU is easy to incorporate into a system since it requires only a single +5V or +3.3V power source, depending on the chosen technology.
All output signals are fully decoded and timed to control standard memory or peripheral circuits; the PE80 CPU is supported by an extensive family of peripheral controllers.
Figure 1 illustrates the internal architecture and major elements of the PE80 CPU.

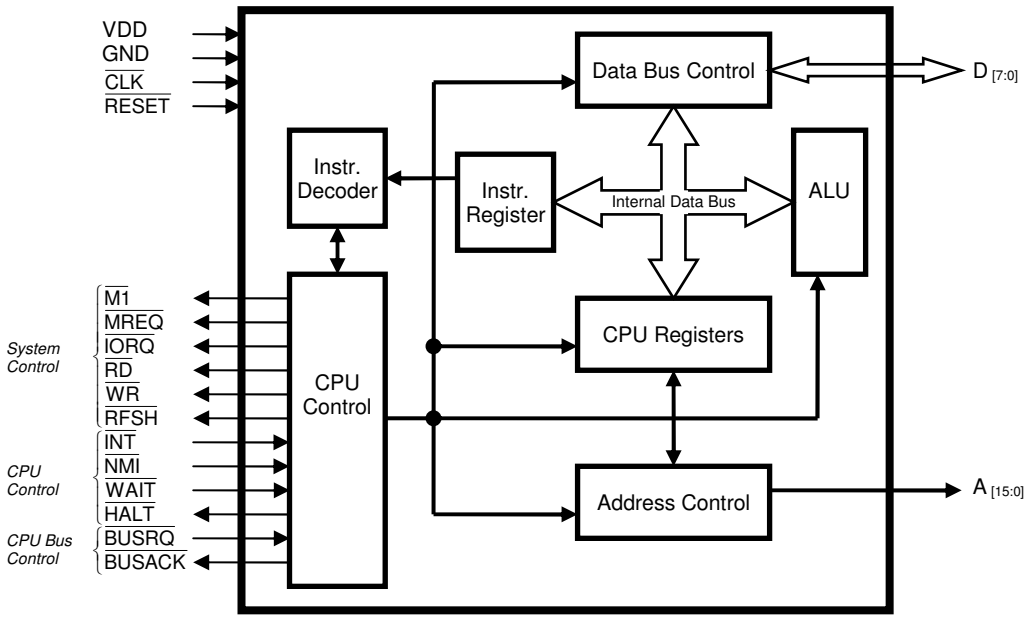

Figure 1: Typical functional block diagram

2. Port Description

A[15:0] *Address Bus (output, active High, tristate).* **$A_{[15:0]}$** form a 16-bit address bus. The Address Bus provides the address for memory data bus exchanges (up to 64 Kbytes) and for I/O device exchanges.

$\overline{\text{BUSACK}}$ *Bus Acknowledge (output, active Low).* Bus Acknowledge indicates to the requesting device that the CPU address bus, data bus, and control signals **$\overline{\text{MREQ}}$**, **$\overline{\text{IORQ}}$**, **$\overline{\text{RD}}$**, and **$\overline{\text{WR}}$** have entered their high-impedance states. The external circuitry can now control these lines.

$\overline{\text{BUSREQ}}$ *Bus Request (input, active Low).* Bus Request has a higher priority than NMI and is always recognized at the end of the current machine cycle. **$\overline{\text{BUSREQ}}$** forces the CPU address bus, data bus, and control signals **$\overline{\text{MREQ}}$**, **$\overline{\text{IORQ}}$**, **$\overline{\text{RD}}$**, and **$\overline{\text{WR}}$** to go to a high-impedance state so that other devices can control these lines. **$\overline{\text{BUSREQ}}$** is normally wired-OR and requires an external pull-up for these applications. Extended **$\overline{\text{BUSREQ}}$** periods due to extensive DMA operations can prevent the CPU from properly refreshing dynamic RAMS.

D[7:0] *Data Bus (input/output, active High, tristate).* **D[7:0]** constitute an 8-bit bidirectional data bus, used for data exchanges with memory and I/O.

$\overline{\text{HALT}}$ *HALT State (output, active Low).* **$\overline{\text{HALT}}$** indicates that the CPU has executed a HALT instruction and is waiting for either a non-maskable or a maskable interrupt (with the mask enabled) before operation can resume. During **$\overline{\text{HALT}}$**, the CPU executes NOPs to maintain memory refresh.

$\overline{\text{INT}}$ *Interrupt Request (input, active Low).* Interrupt Request is generated by I/O devices. The CPU honors a request at the end of the current instruction if the internal software-controlled interrupt enable flip-flop (IFF) is enabled. **$\overline{\text{INT}}$** is normally wired-OR and requires an external pull-up for these applications.

$\overline{\text{IORQ}}$ *Input/Output Request (output, active Low, tristate).* **$\overline{\text{IORQ}}$** indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. **$\overline{\text{IORQ}}$** is also generated concurrently with **$\overline{\text{M1}}$** during an interrupt acknowledge cycle to indicate that an interrupt response vector can be placed on the data bus.

$\overline{\text{M1}}$ *Machine Cycle One* (output, active Low). **$\overline{\text{M1}}$**, together with **$\overline{\text{MREQ}}$**, indicates that the current machine cycle is the opcode fetch cycle of an instruction execution. **$\overline{\text{M1}}$** together with **$\overline{\text{IORQ}}$**, indicates an interrupt acknowledge cycle.

$\overline{\text{MREQ}}$ *Memory Request (output, active Low, tristate).* **$\overline{\text{MREQ}}$** indicates that the address bus holds a valid address for a memory read of memory write operation.

$\overline{\text{NMI}}$ *Non-Maskable Interrupt (input, active Low, edge-triggered).* **$\overline{\text{NMI}}$** has a higher priority than **$\overline{\text{INT}}$**. **$\overline{\text{NMI}}$** is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop, and automatically forces the CPU to restart at location 0066H.

$\overline{\text{RD}}$ *Read (output, active Low, tristate).* **$\overline{\text{RD}}$** indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{\text{RFSH}}$ *Refresh (output, active Low).* **$\overline{\text{RFSH}}$**, together with **$\overline{\text{MREQ}}$** indicates that the lower seven bits of the system's address bus can be used as a refresh address to the system's dynamic memories.

$\overline{\text{WAIT}}$ *WAIT (input, active Low).* **$\overline{\text{WAIT}}$** communicates to the CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter a WAIT state as long as this signal is active. Extended **$\overline{\text{WAIT}}$** periods can prevent the CPU from properly refreshing dynamic memory.

$\overline{\text{WR}}$ *Write (output, active Low, tristate).* **$\overline{\text{WR}}$** indicates that the CPU data bus holds valid data to be stored at the addressed memory or I/O location.

$\overline{\text{CLK}}$ *Clock (input).* Single phase CMOS-level clock.

$\overline{\text{RESET}}$ *Reset (input, active Low).* **$\overline{\text{RESET}}$** initializes the CPU as follows: it resets the interrupt enable flip-flop, clears the PC and registers I and R, and sets the interrupt status to Mode 0. During reset time, the address and data bus go to a high-impedance state, and all control output signals go to the inactive state. Notice that **$\overline{\text{RESET}}$** must be active for a minimum of three full clock cycles before the reset operation is complete.

3. CPU Registers

The PE80 CPU contains 208 bits of R/W memory that are available to the programmer. Figure 2 illustrates how this memory is configured to eighteen 8-bit registers and four 16-bit registers. All PE80 registers are implemented using static RAM. The registers include two sets of six general-purpose registers that may be used individually as 8-bit registers or in pairs as 16-bit registers. There are also two sets of accumulator and flag registers and six special-purpose registers.

- A - 8-bit accumulator
- F - zero, negative, overflow, carry, and BCD flags
- BC - two 8-bit registers, or one 16-bit data/address register
- DE - two 8-bit registers, or one 16-bit data/address register
- HL - one 16-bit address/data register or two 8-bit registers
- SP - stack pointer, 16 bits
- PC - program counter, 16 bits
- IX - 16-bit base register for 8-bit offsets
- IY - 16-bit base register for 8-bit offsets
- I - interrupt vector register, 8 bits
- R - DRAM refresh counter, 8 bits
- Four bits of interrupt status and interrupt mode status

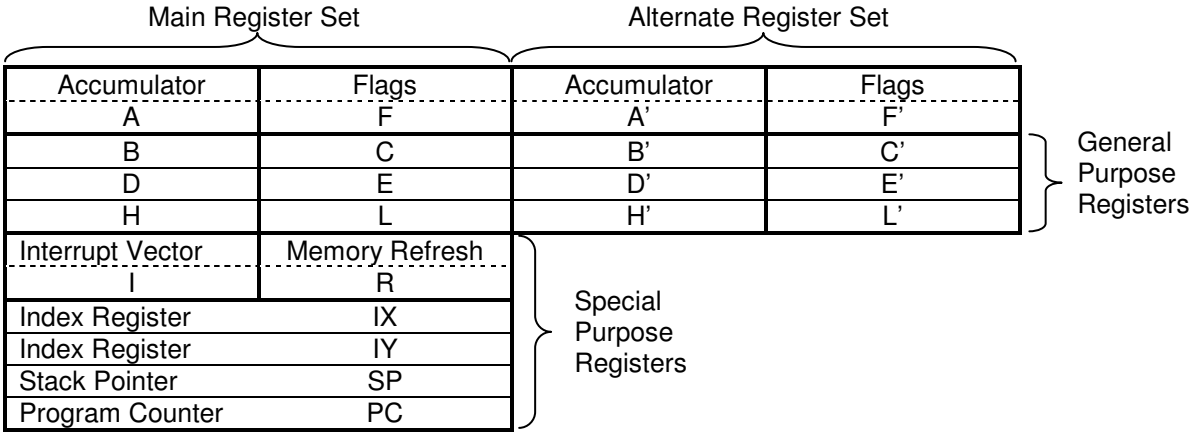

Figure 2: PE80 CPU Register Configuration

3.1. Accumulator and Flag Registers

The CPU includes two independent 8-bit accumulators and associated 8-bit flag registers. The accumulator holds the results of 8-bit arithmetic or logical operations while the FLAG register indicates specific conditions for 8-bit or 1 16-bit operations, such as indicating whether or not the result of an operation is equal to zero. The programmer selects the accumulator and flag pair with a single exchange instruction so that it is possible to work with either pair.

3.2. General Purpose Registers

Two matched sets of general-purpose registers, each set containing six 8-bit registers, may be used individually as 8-bit registers or as 16-bit register pairs. One set is called BC, DE, and HL while the complementary set is called BC', DE', and HL'. At any one time, the programmer can select either set of registers to work through a single exchange command for the entire set. In systems that require fast interrupt response, one set of general purpose registers and an ACCUMULATOR/FLAG register may be reserved for handling this fast routine. One exchange command is executed to switch routines. This greatly reduces interrupt service time by eliminating the requirement for saving and retrieving register contents in the external stack during interrupt or subroutine processing. These general-purpose registers are used for a wide range of applications. They also simplify programming, specifically in ROM-based systems where little external read/write memory is available.

3.3. Special-Purpose Registers

3.3.1. Program Counter (PC)

The program counter holds the 16-bit address of the current instruction being fetched from memory. The PC is automatically incremented after its contents have been transferred to the address lines. When a program jump occurs, the new value is automatically placed in the PC, overriding the incrementer.

3.3.2. Stack Pointer (SP)

The stack pointer holds the 16-bit address of the current top of a stack located anywhere in external system RAM memory. The external stack memory is organized as a last-in first-out (LIFO) file. Data can be pushed onto the stack from specific CPU registers or popped off of the stack to specific CPU registers through the execution of PUSH and POP instructions. The data popped from the stack is always the last data pushed onto it. The stack allows simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data manipulation.

3.3.3. Two Index Registers (IX and IY)

The two independent index registers hold a 16-bit base address that is used in indexed addressing modes. In this mode, an index register is used as a base to point to a region in memory from which data is to be stored or retrieved. An additional byte is included in indexed instructions to specify a displacement from this base. This displacement is specified as a two's complement signed integer. This mode of addressing greatly simplifies many types of programs, especially where tables of data are used.

3.3.4. Interrupt Page Address Register (I)

The PE80 CPU can be operated in a mode where an indirect call to any memory location can be achieved in response to an interrupt. The I register is used for this purpose and stores the high order eight bits of the indirect address while the interrupting device provides the lower eight bits of the address. This feature allows interrupt routines to be dynamically located anywhere in memory with minimal access time to the routine.

3.3.5. Memory Refresh Register (R)

The PE80 CPU contains a memory refresh counter, enabling dynamic memories to be used with the same ease as static memories. Seven bits of this 8-bit register are automatically incremented after each instruction fetch. The eighth bit remains as programmed, resulting from an LD R, A instruction. The data in the refresh counter is sent out on the lower portion of the address bus along with a refresh control signal while the CPU is decoding and executing the fetched instruction. This mode of refresh is transparent to the programmer and does not slow the CPU operation. The programmer can load the R register for testing purposes, but this register is normally not used by the programmer. During refresh, the contents of the I register are placed on the upper eight bits of the address bus.

3.4. Arithmetic-Logic-Unit (ALU)

The 8-bit arithmetic and logical instructions of the CPU are executed in the ALU. Internally, the ALU communicates with the registers and the external data bus by using the internal data bus. Functions performed by the ALU include:

- Add
- Subtract
- Logical AND
- Logical OR
- Logical Exclusive OR
- Compare
- Left or Right Shifts or Rotates (Arithmetic and Logical)
- Increment
- Decrement
- Set Bit
- Reset Bit
- Test bit

3.5. Instruction Register and CPU Control

As each instruction is fetched from memory, it is placed in the INSTRUCTION register and decoded. The control sections performs this function and then generates and supplies the control signals necessary to read or write data from or to the registers, control the ALU, and provide required external control signals.

4. Software verification on PE80 microprocessor

The PE80 design environment based on VHDL allows to easily convert and adapt existing software code that can be read into the ROM and can be executed in a hardware-software-co-simulation to evaluate and verify the correct command execution and peripheral access.

The PE80 is available on FPGA platforms for hardware evaluation and system hardware design for more complex ASICs / SoC development.

For further information on availability and function please contact PE.

Contact Address:

Stuttgart

Productivity Engineering
Process Integration GmbH
Behringstrasse 7
71083 Herrenberg
Germany

Tel.: +49 (0) 70322798 0
Fax: +49 (0) 70322798 29
Email: info@pe-gmbh.com

Dresden Branch

Productivity Engineering GmbH
Sachsenallee 9
01723 Kesselsdorf
Germany

Tel.: +49 (0) 35204777 00
Fax: +49 (0) 35204777 000
Email: info@pe-gmbh.com