



PayloadPlus™ Functional Programming Language

FPL Architecture

The Agere Systems PayloadPlus architecture includes the Fast Pattern Processor (FPP) and the Functional Programming Language (FPL).

The Agere Systems FPP is a network processor IC that handles the wire-speed data path and performs the packet classification and analysis needed, for example, to make routing decisions. The diagram below shows how the FPP fits into a typical line-card configuration.

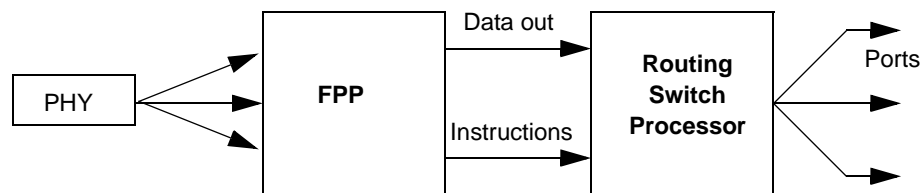
Note that the FPP is devoted to the wire-speed data path. Non-real time functions, such as routing table updates, are handled by a separate external microprocessor. The Agere Systems Routing Switch Processor (RSP) is designed to process FPP outputs, although other application logic can be used.

The Agere Systems FPP is the first network processor IC that can handle complex multi-protocol information at wire speeds up to 2.5 Gbps or OC-48. The FPP can process the

protocols and applications required by carrier and enterprise edge systems. These protocols include IP, ATM, MPLS and AAL5, for example, while the applications include routing, switching, network management, firewalls, monitoring, SARing and Access Control List processing. As a programmable processor, rather than a fixed-function ASIC, the FPP can handle new protocols or applications as they are developed or as users need to add functions to their network.

FPL: High-level Protocol Processing

FPL — the industry's first functional protocol-processing language — is the software driving the FPP. FPL is a very efficient, high-level language that instructs the FPP to perform complex Layer 2 through 7 classification and analysis on the wire-speed data path. Using FPL, system vendors can provide carriers the ability to offer new service offerings via software changes rather than forcing the carriers to replace system hardware.



As a functional rather than a procedural programming language (such as C), FPL requires far fewer lines of code to program a function. The reduction in code is typically about two orders of magnitude, which translates into writing one line of FPL code for every 100 lines of equivalent C

code. Once the basic constructs and syntax of FPL are understood, writing code is highly intuitive. And, with fewer lines of code, it is easier to debug, reuse, and modify.

The Magic of FPL

The key to the extremely high performance of the FPP is in its programming language, FPL. Agere Systems FPL — and the FPP engine that processes it — is implemented using patented algorithms that allow it to function far faster and more efficiently than conventional processing approaches. But how does it run faster? And what makes FPL special, relative to a well-known programming language like C? This section answers those questions.

As we have said, FPL is used to implement protocol processing tasks. Programming in FPL consists of *describing* the protocols and the actions to process them. This contrasts sharply with the conventional approach with a language such as C, where you describe how to process protocols.

Since FPL learns the protocols and how to process them, you don't have to tell FPL what to do every time. You might compare this to the process of tying your shoelaces. You probably learned this skill and simply execute it now without a thought of the independent steps involved. This is the way FPL works — a task is learned, so the overhead of reading, understanding and executing the steps involved in the tasks is eliminated.

Contrast this with a procedural language, which tells the processor what to do. Using the shoelace analogy, you would tie your shoes through a series of complex instructions. Since you never learn the process, the instructions need to have an incredible amount of detail, in the form of subroutines. You might require a different subroutine for each different pair of shoes you own, and perhaps even a dual set of instructions for the right and left shoe. A procedural language never learns, so it must be told what to do, every time.

When you extend this analogy to packet processing, you can see how much processor time is spent just telling the processor what to do *for each packet* that comes down the wire! Not only is the amount of code to execute a procedure much larger, this inefficiency is repeated for every packet. And, because there are so many lines of code, this makes programming, debugging, re-use, and modification a time-consuming and labor-intensive affair.

FPL Terms and Concepts

To effectively write software using FPL, there are a few key terms and fundamental concepts which must be understood. These include:

- *Protocol data unit (PDU)* — a protocol data unit is defined to be any delimited series of data bytes that can be processed by the FPP. This includes frames, packets, or

cells of any size — any protocol, including those yet to be defined.

- *Block* — in FPL, a block is defined as a 64-byte block that is the unit the FPP parses PDUs into for processing.
- *Data stream processing* — the FPL program processes PDUs by processing the data serially, as a data stream.
- *Pattern matching* — as the FPL program processes a data stream, the FPL program looks for bit pattern matches in the PDU to determine the proper processing path.
- *Queue engine* — the FPP features a queue engine that allows it to process PDUs of any size. The PDU blocks are classified and sent to the queue engine. Once a PDU is queued, it can be processed as a whole by the FPP.
- *Replaying PDUs* — the replay is the second-pass processing by the FPP on the PDU input from the queue engine.

Processing PDUs with FPL

By using the Fast Pattern Processor and FPL, you can perform a number of functions on your incoming PDUs, including:

- Layer two and above protocol processing
- Checking programmable PDU size
- Segmentation and reassembly (SAR) of ATM cells
- Timeout checks on ATM cells
- CRC and checksum processing
- Determination of the PDU output queue
- Determination of the PDU class of service

The flexibility of FPL also allows you to define custom functions for manipulating PDUs.

Creating an FPP Program with FPL

FPL allows you to write a program to process PDUs by:

- Defining protocol patterns for the FPP to recognize.
- Defining functions for the FPP to execute based on the pattern recognized — both FPL built-in functions and user-defined functions.
- Dynamically changing — at any time — the patterns to be recognized.

The program is the custom and complete set of functions that control the FPP's handling of PDUs.

For example, the program you write can process a PDU and create conclusions that help by:

- Identifying the output queue.
- Performing functions to process a protocol.

- Replaying the PDU to process more than one protocol for the PDU, if necessary.

The dynamic updating capability allows you to add or delete patterns recognized by the FFP, a useful feature for maintaining such items as IP routing tables.

FFP Two-pass Processing

The FFP is designed to process data streams in two passes:

- An initial processing pass as the PDU blocks are read into the queue engine memory.
- A second processing pass as the PDU is replayed from the queue engine.

The initial pass does preliminary processing that might include identifying the PDU type, reading packet values, and, for ATM, assembling the cells. The second pass, or replay, occurs after all of the blocks for a PDU have been queued. On replay, the program can simply transmit the PDU and conclusions to the application engine, or process a higher level protocol before transmitting.

The FFP Top-level Flow

The FFP operates on 64-byte blocks. The FFP processes the incoming PDU data stream in one or more 64-byte blocks. The 64-byte block and the control information for the PDU are passed to the FPL program. The block is loaded into a buffer that is processed, and the control information is made available to the FPL program as variables. The FPL program can access the block and the control information for the PDU. See the diagram, *FFP Top Level Flow*, that follows.

Loading an FPL Program

The program you write in FPL is compiled to object code, then converted to an image that is loaded into the FFP. This process creates the algorithms that the FFP uses to process PDUs.

Using the Queue Engine

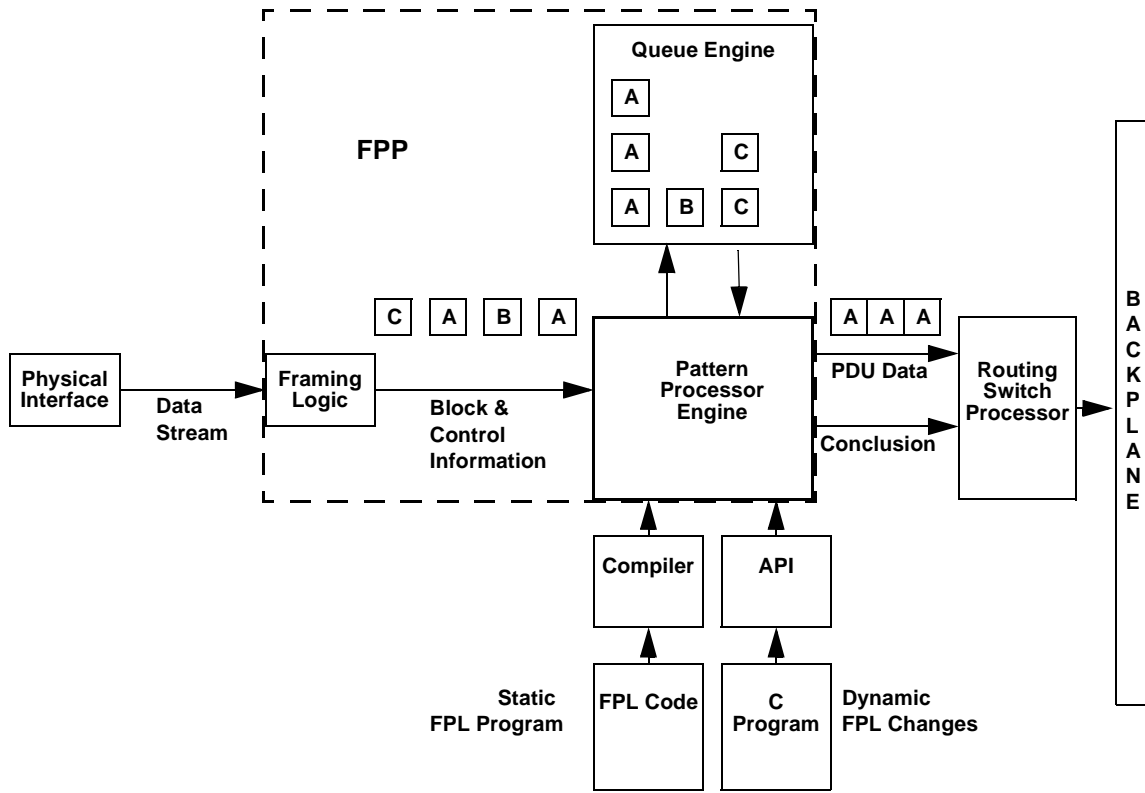
You use FPL to control the flow of the PDU through the FFP queue engine. The FFP queue engine allows you to process PDUs that might be embedded in a higher-level protocol. For example, you can process the PDU once for one protocol, send it to the queue, then process it again for another protocol. In addition, PDUs greater than one block can be stored in the queue for processing.

Making Dynamic FPL Program Changes

You can add and delete certain types of FPL statements from the image dynamically, through the use of an application programming interface (API).

FPL supports two types of pattern statement structures: single-rule pattern statements and multiple-rule pattern statements.

- Single-rule pattern statements have a single pattern to match with one or more functions to perform. These single-rule pattern statements are called *flows*. These statements cannot be added or deleted dynamically.



FPP Top Level Flow

- Multiple-rule pattern statements allow you to define tables, such as IP routing tables, to process a pattern that has variations. These are the type of statements that can be added or changed dynamically at run time. These multiple-rule pattern statements are called *trees*.

To add a new statement, you must add it to the existing program structure. You can only add or delete statements from an existing tree. You cannot create a new flow or tree dynamically; you must create them in the static program.

You dynamically add statements to your FPL program by using the Agere Systems API.

FPL Program Execution Structure

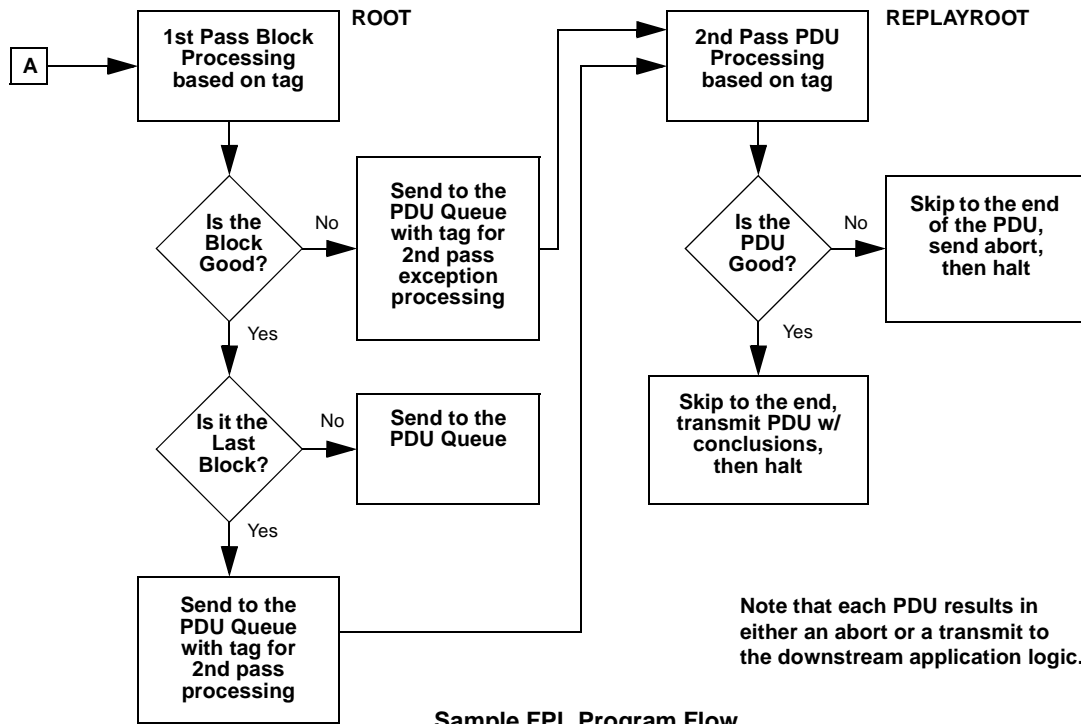
When a PDU arrives at the FPP, the FPL program starts to execute. It continues to run until the PDU is processed. When a new PDU arrives, the program begins again.

A program you create in FPL can have one or two entry points, called *roots*, that roughly correspond to the *main* function of a C program. The root statement defines the starting function for the FPL program when a PDU is processed. The **ROOT** function can receive the data stream

from both the framer and the queue engine. Optionally, you can define a second root, called **REPLAYROOT**, that processes PDUs from the queue engine, allowing the **ROOT** function to process only the information received from the framer.

An FPL program begins with a **ROOT** function that can call other functions, based on the control information and the data pattern of the PDU. A sample program structure might be as shown in the figure that follows.

You can use the FPL tagging capability to define many paths for functions in both first- and second-pass processing to handle different types of data. The first-pass tag is the input port number, and the second-pass tag you assign in the FPL program during first-pass processing.



For additional information, contact your Agere Systems Account Manager or the following:

INTERNET: <http://www.agere.com>

E-MAIL: docmaster@micro.lucent.com

N. AMERICA: Agere Systems, Inc., 555 Union Boulevard, Room 30L-15P-BA, Allentown, PA 18103

1-800-372-2447, FAX 610-712-4106 (In CANADA: **1-800-553-2448**, FAX 610-712-4106)

ASIA PACIFIC: Agere Systems, Inc., Singapore Pte. Ltd., 77 Science Park Drive, #03-18 Cintech III, Singapore 118256

Tel. (65) 778 8833, FAX (65) 777 7495

CHINA: Agere Systems, Inc. (China) Co., Ltd., A-F2, 23/F, Zao Fong Universe Building, 1800 Zhong Shan Xi Road, Shanghai 200233 P. R. China **Tel.**

(86) 21 6440 0468, ext. 316, FAX (86) 21 6440 0652

JAPAN: Agere Systems, Inc. Japan Ltd., 7-18, Higashi-Gotanda 2-chome, Shinagawa-ku, Tokyo 141, Japan

Tel. (81) 3 5421 1600, FAX (81) 3 5421 1700

EUROPE: Data Requests: Agere Systems, Inc. DATALINE: **Tel. (44) 7000 582 368**, FAX (44) 1189 328 148

Technical Inquiries: GERMANY: **(49) 89 95086 0** (Munich), UNITED KINGDOM: **(44) 1344 865 900** (Ascot),

FRANCE: **(33) 1 40 83 68 00** (Paris), SWEDEN: **(46) 8 594 607 00** (Stockholm), FINLAND: **(358) 9 4354 2800** (Helsinki),

ITALY: **(39) 02 6608131** (Milan), SPAIN: **(34) 1 807 1441** (Madrid)

Agere Systems, Inc. reserves the right to make changes to the product(s) or information contained herein without notice. No liability is assumed as a result of their use or application. No rights under any patent accompany the sale of any such product(s) or information.

Copyright © 2001 Agere Systems, Inc.
All Rights Reserved
Printed in U.S.A.

5/15/01
Document Number



