Maxeler Apps
HTTP Web-Server

MAXELER
Technologies
MAXIMUM PERFORMANCE COMPUTING

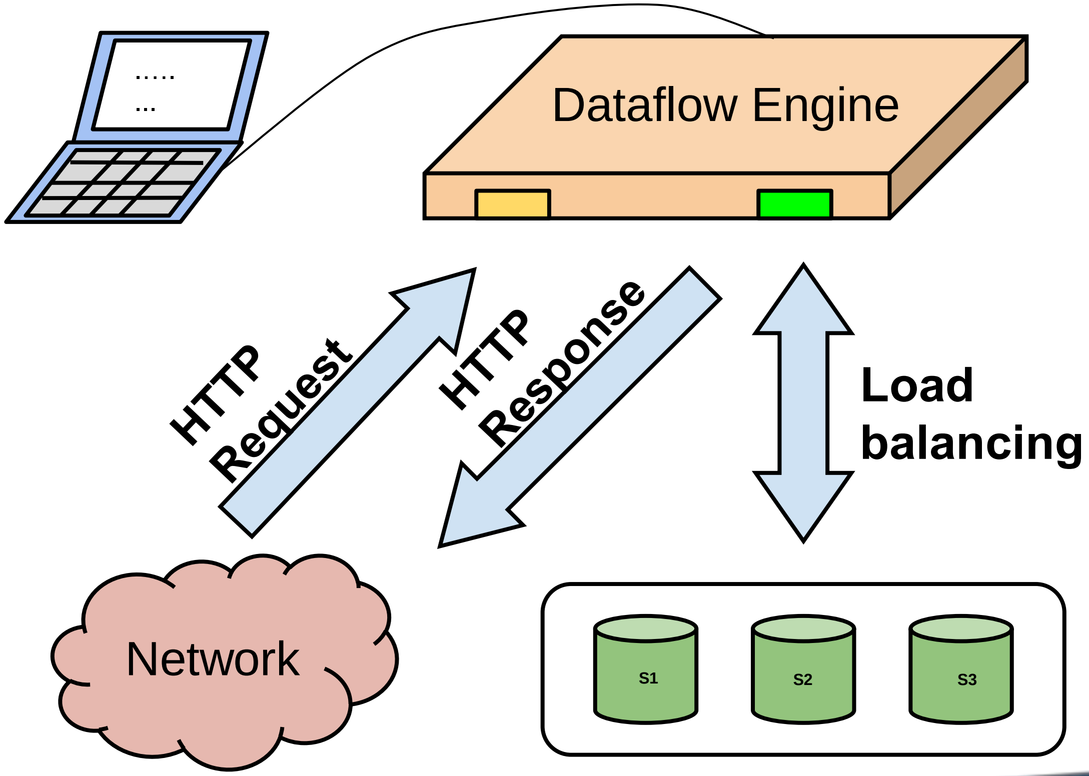
Jan 2015

HTTP Web-Server

- A web server is a computer system that processes requests via HTTP network protocol used to distribute information on the World Wide Web. The most common use of web servers is to host websites.
- The primary function of a web server is to store, process and deliver web pages to clients. The communication between client and server takes place using the Hypertext Transfer Protocol (HTTP). Pages delivered are most frequently HTML documents, which may include images, stylesheets and scripts in addition to text content.
- A user agent, commonly a web browser or web crawler, initiates communication by making a request for a specific resource using HTTP and the server responds with the content of that resource or an error message if unable to do so. The resource is typically a file on the server's storage, but this is not necessarily the case and depends on how the web server is implemented. If specific resource is not on the server, one possible solution would be to forward incoming request to the other servers using a user defined load balancing algorithm.

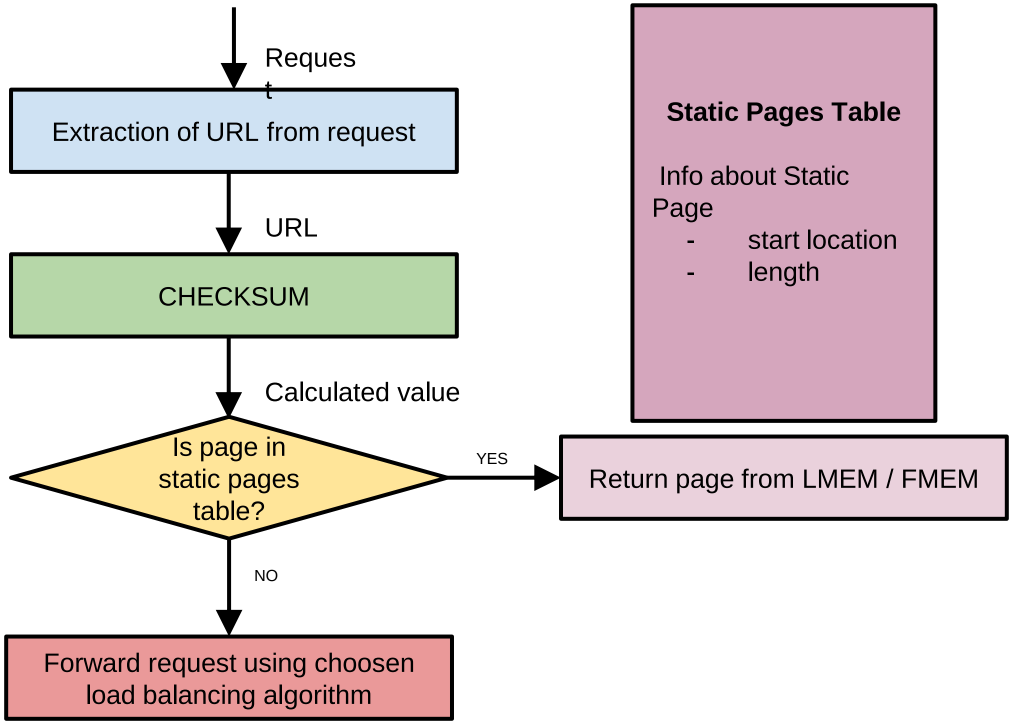
HTTP Web-Server

- For DFEs with Ethernet connectors (MAX4N, JDFE)
- Handles incoming HTTP requests
- If web pages exist in DFE LMEM, returns HTTP Responses
- In case web pages are not on the DFE, forwards incoming request to the other servers using a user defined load balancing algorithm



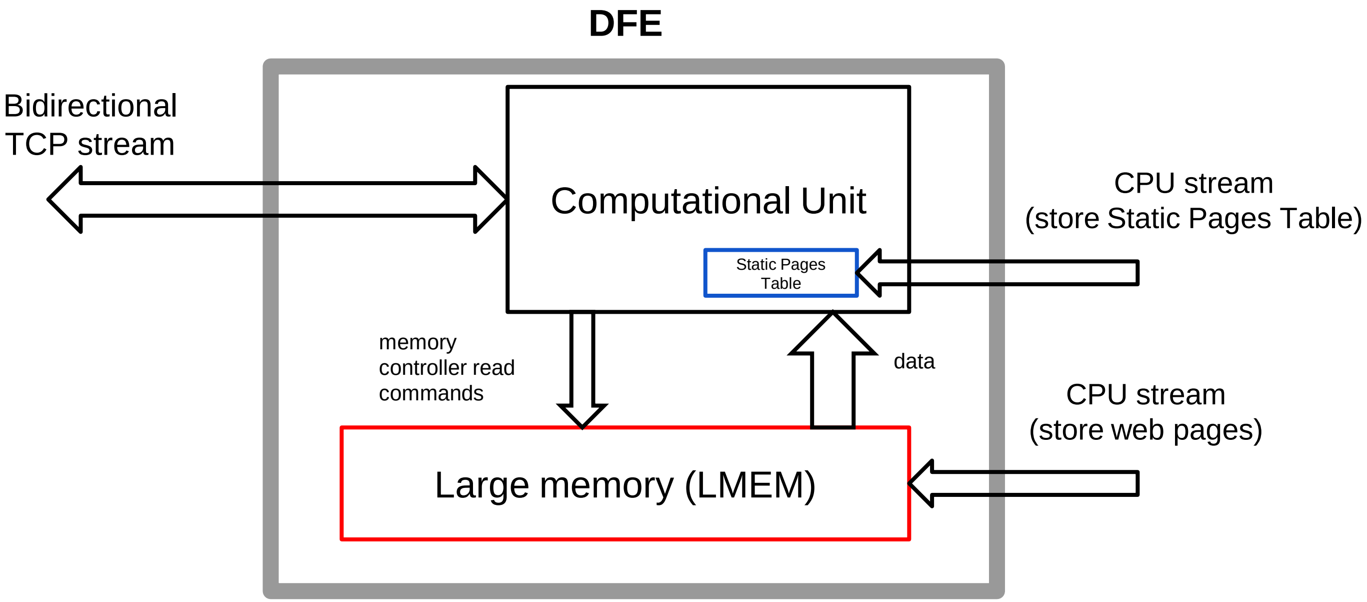
HTTP Web-Server

- Client sends HTTP request
- DFE has a TCP-IP core and extracts URL from HTTP request, calculates URL's checksum value and checks Static Pages Table
- If page exist, Server returns page from LMEM, otherwise Server forwards request using a user-defined load balancing algorithm


Reques
t
Extraction of URL from request
URL
CHECKSUM
Calculated value
Is page in static pages table?
YES
Return page from LMEM / FMEM
NO
Forward request using choosen load balancing algorithm
Static Pages Table
Info about Static Page
- start location
- length

Inside the DFE

- HTTP request/HTTP response received/sent via Bidirectional TCP stream
- New web pages (and the updated table) are pre-loaded in LMEM via the CPU stream
- Computational unit generates memory controller read commands and receives LMEM data

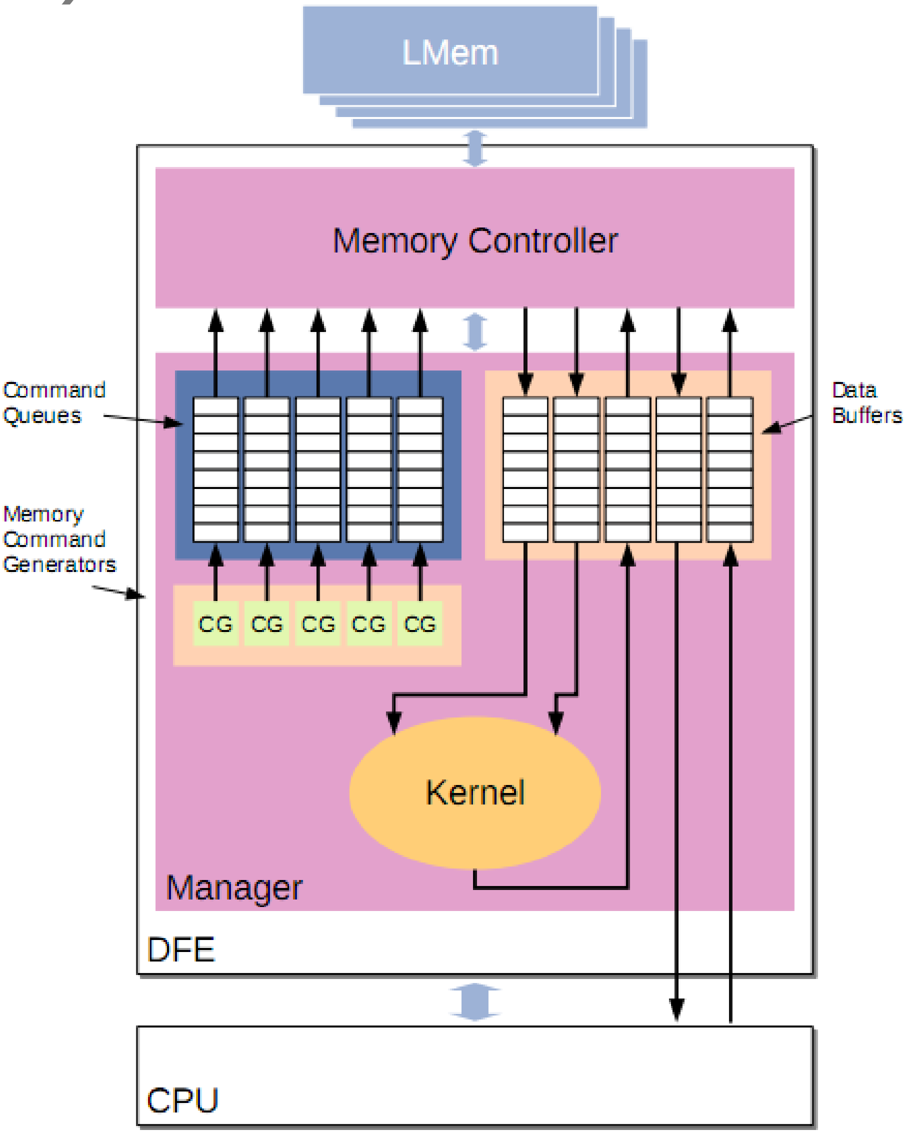

DFE
Bidirectional TCP stream
Computational Unit
Static Pages Table
CPU stream (store Static Pages Table)
memory controller read commands
data
CPU stream (store web pages)
Large memory (LMEM)

Computational unit

- Receives TCP data from network interface
- Detects HTTP request and extracts URL
- Calculates checksum (of URL) and obtains checksummed index
- From checksummed index table retrieves start address and burst number of a requested page
- Sends read commands to LMEM until entire page/request is read from the LMEM
- Receives LMEM data and sends HTTP response containing requested page back to a client

Large Memory (LMEM) Access

- To read data from Lmem read command must be sent to the memory controller
- Data is read from LMem in bursts - blocks of bytes (e.g. burst=192 B)
- Multiple read commands queued (buffer)



Code example

- Computational code that generates write/read commands for LMEM memory controller.
- Used when state machine needs to access LMEM.
- It is assumed that data streams are connected to LMEM.
- Commands are generated by using the following method:

```
private void makeReadOrWriteCommand(DFEsmStateValue command,
                                    DFEsmValue address, int streamIx)
```

Parameters:

command - after function's call, it contains new read/write command

address - LMEM burst number; user specifies where to/from write/read data

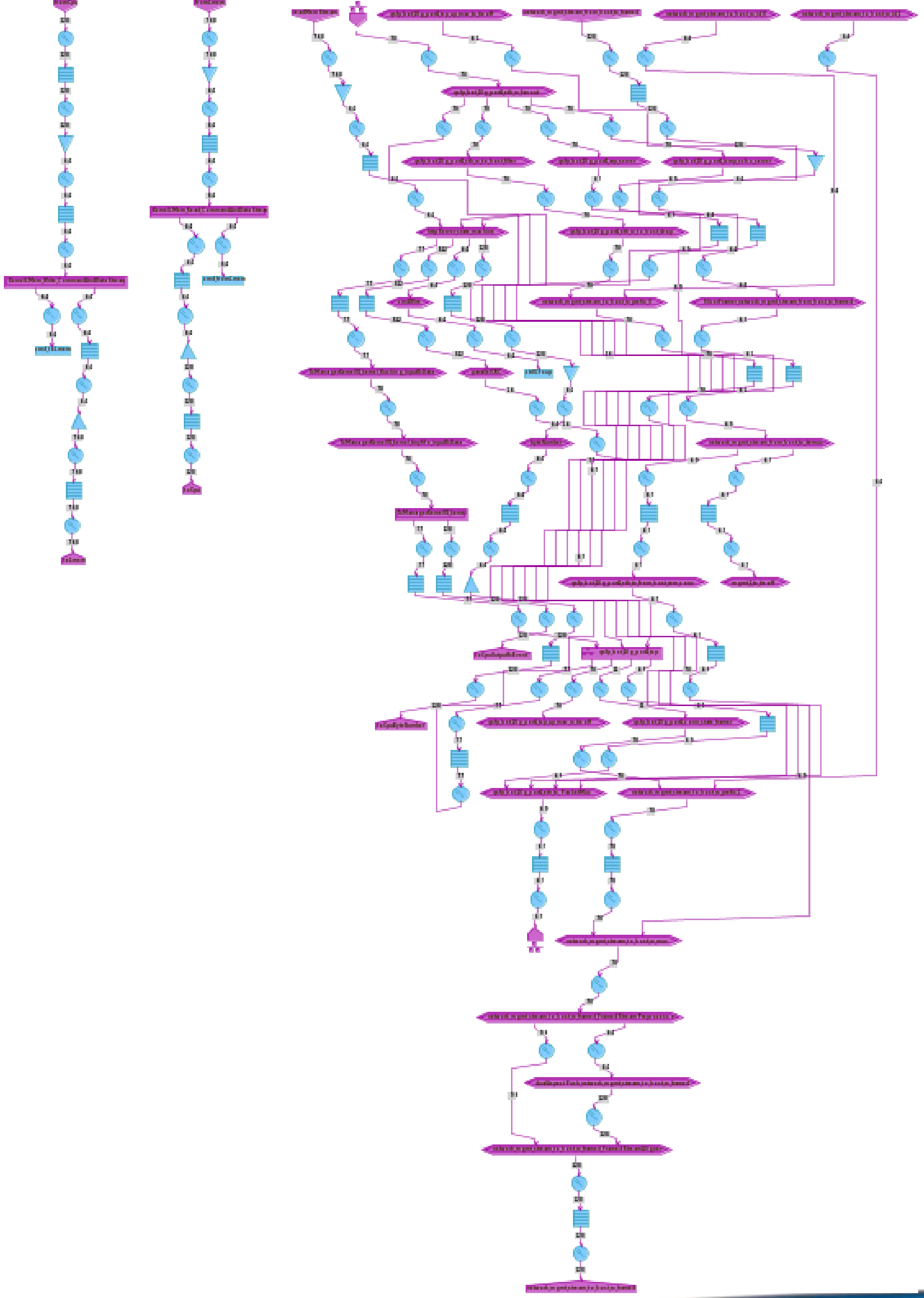
streamIx - LMEM data stream ID number;
user specifies desired data stream connected to LMem

Code example

- Here is an example how to generate the read commands within the State Machine:

```
IF(sMemoryRead === true) {  // If signal to read LMEM is set
    IF(~memCmdOutput.stall) {  // If LMEM command stream output is not in stall
                // On each cycle, increase the counter;
        // It counts until all LMem bursts are read
                sLMemBurstCounter.next <== sLMemBurstCounter + 1;
                // Generate the read command
                MakeReadOrWriteCommand(
          memCmdReg,  // Contains the read command
                  (sStartBurstAddress + sLMemBurstCounter).cast(dfeUInt(32)),  // Specify burst to
read
                  readStreamID  // Specify LMEM data stream
          );
                memCmdRegValid.next <== true;  // Set the output's variable to true
                ...
  }
}
```

Manager graph



Resource Usage

- Logic utilization: 76752 / 359200 (21.37%)
- Primary FFs: 119372 / 718400 (16.62%)
- Secondary FFs: 5789 / 718400 (0.81%)
- Block memory (M20K): 782 / 2640 (29.62%)
- DSP blocks: 2 / 352 (0.57%)
- Clock Frequency: 100 MHz

DFE Performance - Latency

- TBD