Processing of network traffic in low-power devices with FPGA

Jan Viktorin

Low-power processing of network traffic is an important topic nowadays. There are many different devices running non-stop without human intervention like routers, switches, probes, etc. Such devices can take advantage of an FPGA for high-speed processing while preserving low power consumption. This leads to designing solutions with tightly integrated CPUs and FPGA on the same die. Such architectures enable moving of data processing from CPU to a hardware accelerator running in FPGA dynamically on demand, for example, when the system load becomes too high. Higher performance while reducing power consumption is obtained as the hardware accelerator can process more data compared to the CPU in the same time period.

Consider a system with an input buffer of capacity *n* and capacity in a certain time n_t ($n_t < n$), a data source of a throughput in a certain time s_t and its maximal throughput s_{MAX}, and two processing elements of throughputs r_{SW} (software) and r_{HW} (hardware). It is possible to determine the speed of filling the buffer n_t' (derivative of n_t) as $n_t' = s_t - r$ where r is either r_{SW} or r_{HW} depending on the current system state. It is possible to switch processing throughput from r_{SW} to r_{HW} and vice versa. The system spends (constant) time T_{off} on switching processing elements from software to hardware. The opposite direction is not to be studied in this work. Thus, if $n_t' \cdot T_{off} = n_t$ the system would switch the processing elements without overfilling the input buffer (no data loss). The worst case throughput on input is when $s_t = s_{MAX}$ and it determines the earliest time point when the switch of processing elements makes sense. However, switching processing elements at this time point is not optimal because the network traffic is dynamic in time. For example when $s_{t_0} > s_{t_1}$ for $t_0 < t_1$ there is no need to spent time (power) on the switching of processing elements.

To improve the determination of a good time point to switch processing elements, we can design a predictor of n_{t+k}' (that is a predictor of s_{t+k}) for $k > 0$. A good predictor can delay the switching until it is certainly needed. For that purpose, it is desirable to have a system that helps to analyze behaviour of a set of predictors on a real or fictive network traffic. In this work, I want to specify a dynamic system for data processing by means of Petri nets to be able to simulate a network traffic processing system.