Subject: Re: Ptolemy II perfomance
Posted by Krzysztof Korcyl on Tue, 25 May 2004 10:18:32 GMT
View Forum Message <> Reply to Message

Dear Sergey,

I need a bit more clarifications on the test with systemC with chain of delays. I do not know the systemC but from the MainProgram.cpp I am guessing the following (big capital letters are comments to the MainProgram.cpp code):

A. system is composed of one generator,

```
TGenerator* generator = new TGenerator("Generator", 1.);
```

B. there is a variable number of delays, where each consecutive delay produces shorter delay (range from 0.999 - 0.899). The delay object gets a message on it's input and relays it to it's output after it's internal delay.

```
TTimedDelay* delays[numdelays];
for (int n=0;n<numdelays;n++)
  delays[n] = new TTimedDelay(mname("D",n), 0.999 - 0.1*n/numdelays);
```

C. There is one sink in the system which deletes all messages arriving on it's input

```
TDiscard* discard = new TDiscard("Discard");
```

D. The delay objects are connected via another objects: token_channel. Each token_channel receives a message on it's input and relays it on it's output after a fixed delay - which is apprently 0.0 (ie the token_channel is infinitively fast). I understand, that this is systemC requirement to use such token_channels to connect objects (in general one can assign a non zero delay to the token_channel). It that correct?

```
TToken_channel* chanels[numdelays+1];
for (int n=0;n<=numdelays;n++)
  chanels[n] = new TToken_channel(mname("C",n), 0.);
```

E. Below is code making connections between delay objects and token_channels

```
generator->output(*chanels[0]);
for (int n=0;n<numdelays;n++) {
  delays[n]->input(*chanels[n]);
  delays[n]->output(*chanels[n+1]);
}
```

F. Here you end the chain with discard object.

```
discard->input(*chanels[numdelays]);
```

The simulation operates as follows:
Generator produces a message and passes it into the first token_channel. The first delay gets the message and after it's delay puts it on output. The second token_channel gets the message and passes it immediately to the next delay object and so on. The minimum inverval

between two consecutive messages from the generator is greater (1.0) than maximum delay (0.999), thus we will never have problem of buffering message due to some channel being occupied by previous message.

What was your measure when running the simulation: number of messages generated by the Generator?

cheers,

Krzysztof
