Open62541 publishing performance on a Soft-Core CPU in an FPGA

11. February 2020

In the first steps to get OPC UA PubSub running on MicroBlaze Soft-Core the performance was not in the focus. At the TSN/A conference in October 2019 was an interesting presentation about the processor cycles for the publisher. It was known that this is a bottleneck, especially on low performance CPUs. Therefore, an improvement with some static configuration was presented on the roadmap. At the SPS in Nurnberg we saw first figures with this performance improvement. Now it will be very interesting what does this mean on a low performance CPU like the Xilinx MicroBlaze Soft-Core in the FPGA. If this performance is sufficient, a smart combination with NetTimeLogic's TSN products and an open62541 PubSub application in a MicroBlaze Soft-Core will fulfill many market requirements.

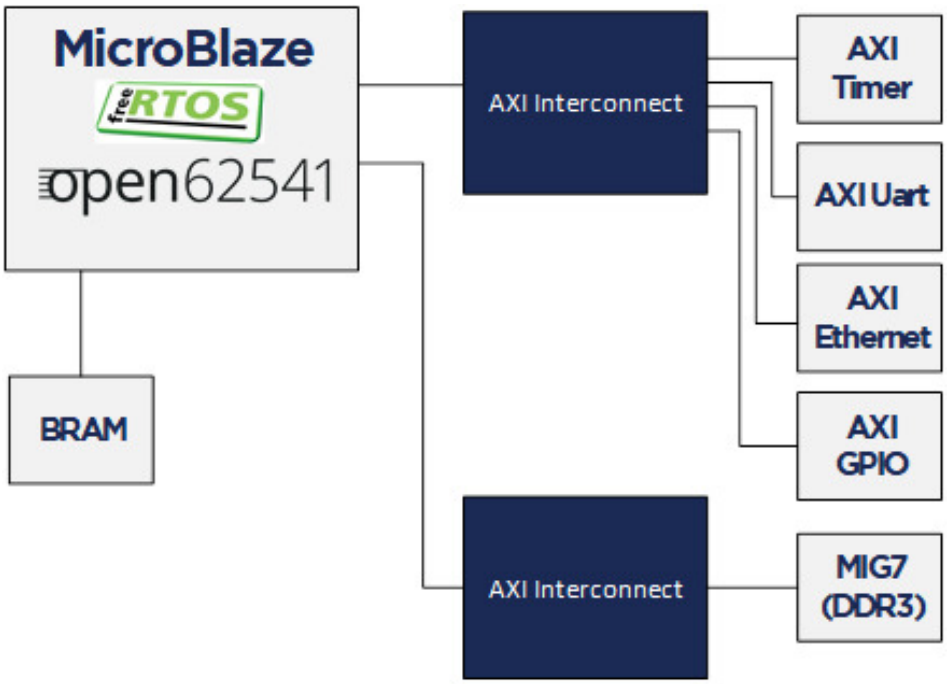
As a starting point, the OPC UA PubSub tutorial on a FPGA is used (OPC UA PubSub on a FPGA using open62541). To get the latest RtLevel features of the open62541 implementation the master branch is used.

The example FPGA project and the application are available here:
https://github.com/NetTimeLogic/opcua/tree/PubSub_RtLevel_example

The open62541 implementation is available here (master):
https://github.com/open62541/open62541/tree/master

Introduction

The main change compared to the previous open62541 posts is that now the master branch of open62541 is used. There were some small adjustments needed in the application code, otherwise the old tutorial on how to generate the libraries is still valid.
Also the MicroBlaze FPGA image running on the Arty A7-100T development board from DIGILENT is still unchanged.



The main focus is to compare the publish performance of frames with the different PubSub RT levels. Deterministic behavior was not investigated and also not the PubSub conformance (OPC UA part 14) since the feature is still under development.

Description of the RT Modes:

UA_PUBSUB_RT_NONE

Default "none-RT" Mode. For each DataSetField the value is read out of the information model. This is slowing down the publishing process.

UA_PUBSUB_RT_DIRECT_VALUE_ACCESS

Within this RT-mode, the value source of each field is configured as static pointer to a DataValue. The publish cycle is improved by prevent the value lookup within the information model. All fields must be configured with a static value source. The DataSetFields can still have a variable size. The published fields are not visible in the information model.

UA_PUBSUB_RT_FIXED_SIZE

All DataSetFields have a known, non-changing length. The server will pre-generate some buffers and use only memcopy operations to generate requested PubSub packages. The configuration must be frozen while it is operational. The published fields are not visible in the information model.

Design preparation

For the detailed design preparation steps please check the post OPC UA Server on a FPGA using open62541 and OPC UA PubSub on a FPGA using open62541.

Basic OPC UA Server PubSub application

In the Xilinx SDK the available OpcServer.c can be imported to the OpcServer application project.
In the basic server the thread stack size was defined with 4096. This is not enough anymore and the application will report with the hook functions a StackOverFlow. Therefore, the THREAD_STACKSIZE was increased to 16384.
In a first step the network initialization and the basic OPC UA Server configuration is done. Before the server starts, the PubSub specific setup is needed. The application is targeted to be compatible with the Pub/Sub format for the IIC TSN Testbed interoperability application.
With different defines the tested configurations can be selected.

Changes compared to the last version

For the new PubSub feature some changes compared to the previous version are needed.

On the open62541 repository are examples available how the new feature can be used. With the help of the following examples the OpcServer.c was extended.
https://github.com/open62541/open62541/blob/master/examples/pub-sub/server_pubsub_publisher_rt_level.c

Beside some small cosmetic adjustments and some added defines to compile the different versions mainly three adaptations were needed. The first one is how the data set field is added (addDataSetField function), then how the data set value is updated (valueUpdateCallback) and the last one is the freeze of the writer group before publishing starts (UA_Server_freezeWriterGroupConfiguration).

addDataSetField / UA_Server_addDataSetField:
The field needs to be set as static value source and the value must be assigned. A dynamic node is not added as before because of the pointer assignment of the published value.

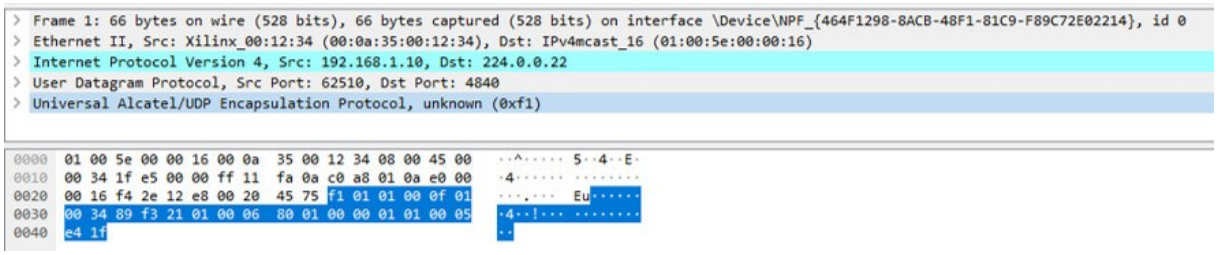
valueUpdateCallback:
The updates of the values is done here (Pointer to the value).

UA_Server_freezeWriterGroupConfiguration:
This must be done before the publishing starts (only for the RT modes). It means that no dynamic changes of the write group are possible anymore. Before any change can be done, publishing must be stopped and the command UA_Server_unfreezeWriterGroupConfiguration called.

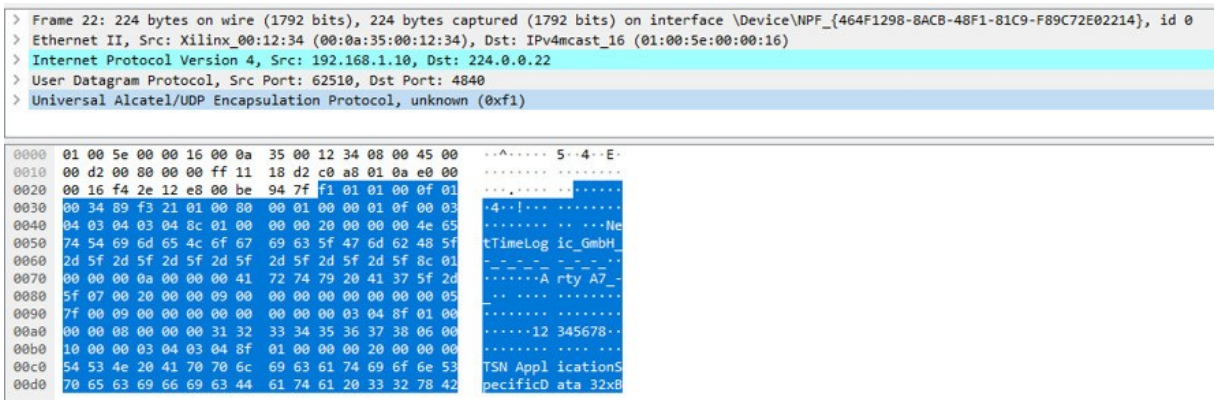
Measurements

As already mentioned in the beginning, the tests are only focusing on how many frames can be published. For all tests the same pub sub message was used and the measurement was done over 180 seconds. The MicroBlaze Soft-Core is running on 100MHz. No other connections to the server (e.g. disconnect UA Expert) are established.
The measurement was done with three different payload configurations for the three different modes. The header was always the same (Timestamp deactivated).

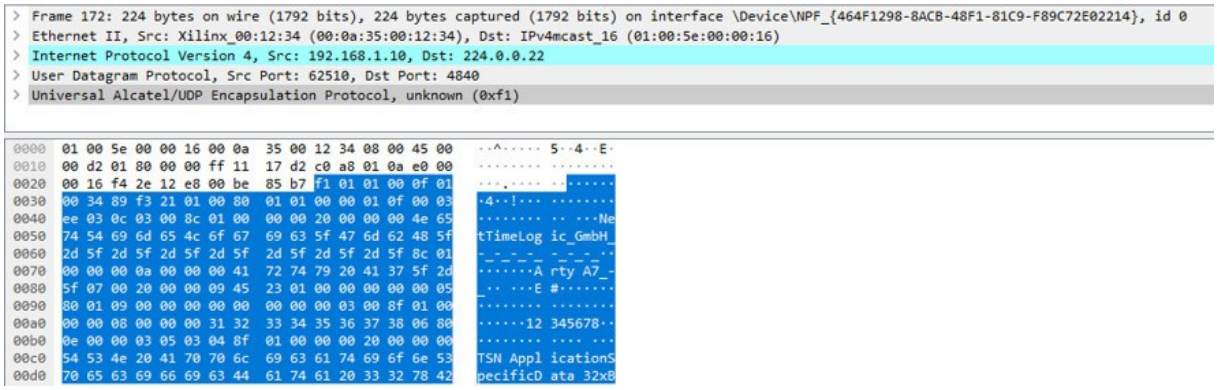
1. 1 published variable, with 1 dynamic value (payload: 24 bytes)



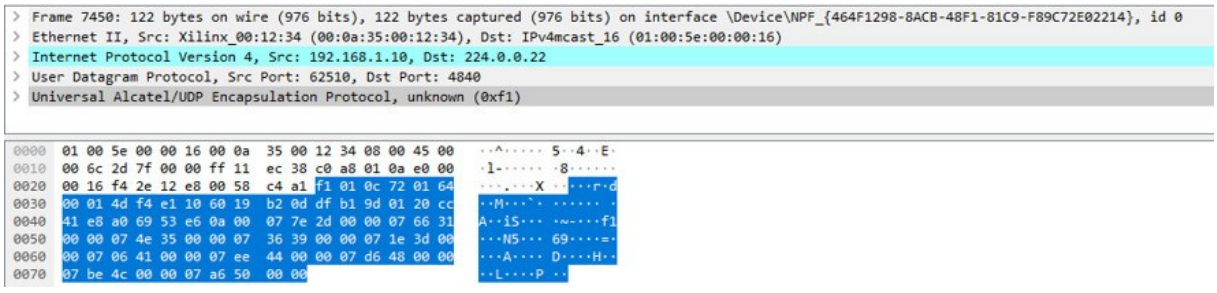
2. 15 published variables, with 1 dynamic value (payload: 182 bytes)



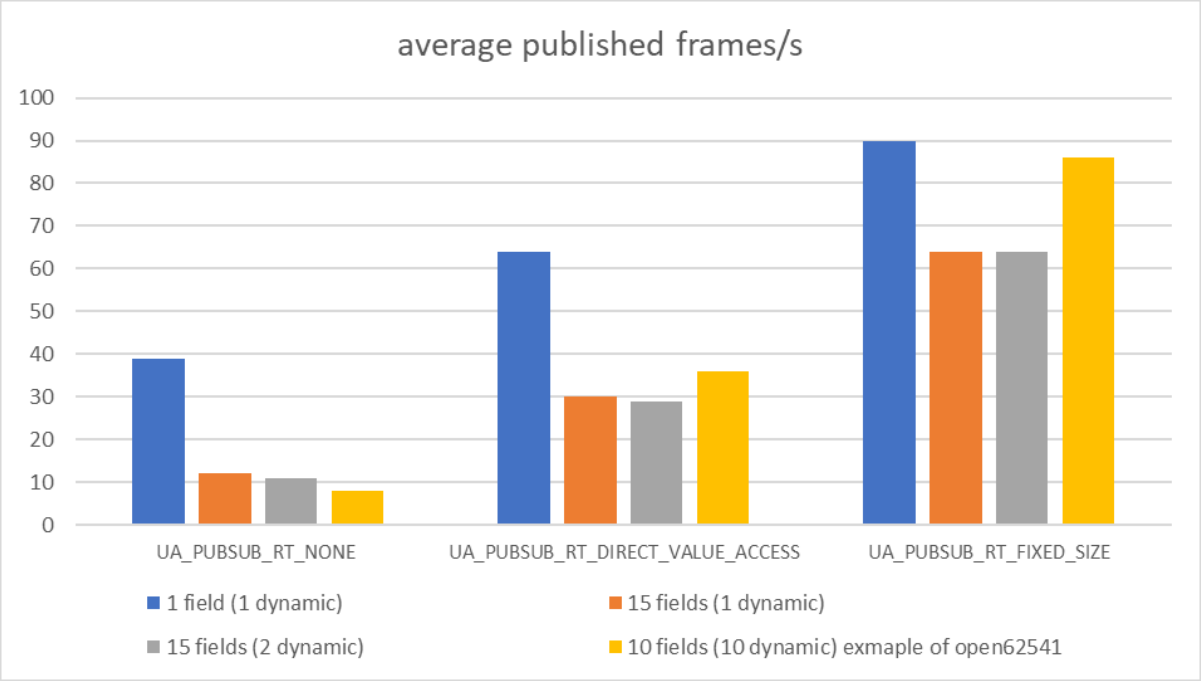
3. 15 published variables, with 2 dynamic values (payload: 182 bytes)



As a fourth frame format the example [RT level PubSub application from open62541] was also tested. This has 10 published variables and all are dynamic (payload 80 bytes)



Measurement overview:



	1 field (1 dynamic)	15 fields (1 dynamic)	15 fields (2 dynamic)	10 fields (10 dynamic) exmaple of open62541
UA_PUBSUB_RT_NONE	39	12	11	8
UA_PUBSUB_RT_DIRECT_VALUE_ACCESS	64	30	29	36
UA_PUBSUB_RT_FIXED_SIZE	90	64	64	86

Observed problems with the RtLevel UA_PUBSUB_RT_FIXED_SIZE:

- Header information update seems no to work as for the other modes (e.g Seq.-Nr).
- With UA_UADPNETWORKMESSAGECONTENTMASK_TIMESTAMP enabled the dynamic value does not update anymore. Therefore, all tests were done without this flag. The reason was not investigated.

Summary

The RT level PubSub update of open62541 brings a substantial performance improvement already on a very low performant CPU. With the enhancement, also some limitations were added. Dynamic changes of the published Datasets are not possible while it is operational. This is most probably negligible for many applications. Another constraint of the RT level is that the values are not visible in the

information model anymore. Updating the information model would decrease again the performance.
Especially the RT_FIXED_SIZE concept does show the benefit when many fields are published. It seems definitely as an option to use open62541 in combination with a Soft-Core in the FPGA. Of course, high-performance applications can't be achieved, but for such cases usually a high perfmance CPU is anyhow already in place for the application part.
However, there are still some open points when it comes to dynamic fields of the header like timestamps or sequence number. There seems to be some remaining work (Updates in the header in the UA_PUBSUB_RT_FIXED_SIZE mode).
Some general updates in the header seems not to work as expected:

GroupHeader

- GroupVersion was not assigned
- SequenceNumber was not assigned

ExtendedNetworkMessageHeader

- Timestamp was empty

For our testing we have done the same quick fix as described in the previous post.
As a next step the deterministic behavior will be the focus. Preferable already in combination with our TSN End Node.