29th Congress of the International Council of the Aeronautical Sciences
St. Petersburg, Russia; September 7-12, 2014

TOOLSET FOR ONBOARD NETWORKS DESIGN AND CONFIGURATION

Yuriy Sheynin*, Elena Suvorova*, Alexey Syschikov*, Boris Sedov*, Nadezhda Matveeva*, Dmitry Raszhivin*
*** Saint-Petersburg State University of Aerospace Instrumentation**
Saint-Petersburg, Russian Federation

Keywords: *SpaceWire network, automated design, design space exploration, SpaceWire configuration*

Abstract

SpaceWire networks design and configuration is a complicated design problem that includes a set of different tasks to be solved: terminal nodes selection, workload tasks mapping, logical channels definition, interconnection topology design and switches selection, configuration space settings and routing table generation, etc. In design one should take into account many characteristics and requirements: terminal nodes performance for workload, channels latency and throughput, communication system capacity to cover logical channels requirements, etc. Every design step and the whole network should also fit into other user and technological constraints: area, mass, power consumption, heating also.

Design tasks are complicated; some of them are NP-hard. "Manual" design of SpaceWire networks with dozens of nodes becomes very complicated, time-consuming and has high probability of design errors and misses.

To deal with the problem a through design flow for SpaceWire Networks Design and Configuration has been developed. Toolset includes network architecture design tool, communication structure forming tool and tool that forms logical structure and routing information. All these tools are highly automated and produce results corresponding to the defined user requirements and technological constraints. Network modeling tool is also provided for simulation of network functioning and collection of network statistics for investigation purposes.

1 Introduction

A network synthesis problem for systems with dozens and hundreds of nodes in general case is NP-hard. Nowadays in a number of network synthesis methodics affordable computation complexity is reached due to significant constraints in the problem statement only.

Some input parameters for SpaceWire network generation are determined at the architecture stage on the base of tasks parameters. Tasks should be allocated to nodes. The network should transmit data packets between source and destination terminal nodes with required throughput and timing parameters.

In system level design a network should be generated of devices (routers) from a system components library. The system components library can include different types of routers with various number of ports, mass, power, timing parameters.

The network should also meet user requirements in total equipment mass and power, in fault tolerance, considers specific requirements based on spatial placement of nodes.

Network synthesis methodic, such as based on Stainer three synthesis [1,2,3,4] and other classical methodics for wireless networks synthesis [1,5,6], do not deal with these constraints and can't be used for SpaceWire networks synthesis.

Development of a SpaceWire network structure correspondingly to architecture and others user's requirements are supported by the suggested methodology and developed tools.

2 Architecture

The system design techniques and supporting software are based on the design space exploration methodology. This approach is productive and is widely used in modern R&D. For example, communication architectures with big number of channels generate exponentially growing number of possible mappings of logical channels to the communication paths in the physical structure of the communication network. It makes impossible a straightforward exhaustive exploration of variants. Interrelation between possible mappings and communication protocols characteristics increases complexity of actions in design space exploration.

The proposed instrument is an automated tool for the design of SpaceWire networks.

The methodology of a network structure design includes four main stages:

- Analysis of the workload for the network to be design;
- Synthesis of the network architecture, including the terminal nodes and logical channels;
- Synthesis of the communication system, including switches and physical channels.
- Synthesis of the Logical Network structure and correspondent setting in nodes and switches

Design of SpaceWire network architecture is performed from a given set of basic system components from a network system components library according to user-defined network characteristics.

During forming of the SpaceWire network architecture it is necessary to verify the possibility of allocation of computation workload to the particular library of system components. To achieve this, the allocation of computing workload to synthesized architecture should be done.

Formed structure should correspond to the user requirements: performance, throughput, command transmit delay, signal of real time transmit delay, failure tolerance of the network, etc.

The problem is solved with a restricted search. Methodology generates a set of possible solutions, which includes alternative problem solutions. Search is restricted by using a number of criteria for discarding not satisfying solutions at each stage.

The assignment problem for the network synthesis has a task graph G as the input data.

$$G = \{N, E\}, \quad \text{where}$$

N – set of graph vertexes, which are computation tasks,

E – set of graph edges.

Additionally we define the overall throughput of the network input/output ports (T) in each direction (for input and for output (I)).

$$\forall t \in T = \sum I_q * I_t$$

Task graph vertexes are mapped to the system library components and a set of possible solutions is formed. At this stage the job is to define the number of nodes of each type and allocate task graph vertexes to the SpaceWire network nodes (end nodes). The total network characteristics should not override the user-defined constraints.

The allocation of the task graph vertexes on the network nodes is represented by the

$$\text{Allocation} = \{\text{A}\}, \text{ where}$$

$$\forall \text{i} \le \text{k} \;\; \text{a}_\text{i} \in \text{A} = \text{Allocation(i)} = \\ = \{\text{n}_\text{i} \in \text{N}, \; \text{n}_b \in \text{N}_b\},$$

where

k – total number of nodes in the task graph;

n_i – node of the task graph allocated to the node n_b of the network.

The function for allocation of a task graph vertex to the network node is:

$$\text{alloc}(\text{n}_\text{i}) = \text{Allocation(i)}$$

The function of a network node allocation that defines which task graph vertexes are allocate to this network node:

$$place(b) = \bigcup Allocation(n): Allocation(n) = b$$

Formula for the total weight of the SpaceWire network is:

$$W = \sum_{i=1}^{k} w_i, \qquad where$$

k – the total number of nodes in the network;

$w_i \in W_b$ – weight of the i-the node of the network.

To limit the search space we apply the criteria:

1. The total weight of the SpaceWire network limit.
$$W \leq W_c$$
2. The limit of the throughput of output logical channels of the network node. The required output logical channels throughput is:
$$T_o = \sum e_t$$
The criterium is:
$$T_o \leq \sum_{n=1}^{k} T_n, \qquad where$$
T – throughput of input/output ports of the network node.
3. The limit of the throughput of input logical channels of the network node. The required input logical channels throughput is:
$$T_i = \sum e_t, \ where$$
The criterium is:
$$T_i \leq \sum_{n=1}^{k} T_n$$
4. The limit of the requirement to the network node memory.
$$\forall n_b \in N_b, \sum_{n \in place(n_b)} m_n \leq m_{n_b}$$
5. The limit of the requirement to network node computation resource (processor)
$$\forall n_b \in N_b, \sum_{n \in place(n_b)} f_{n_b} \leq 100$$

In the process of possible solutions building the best ones are selected with the complex set of minimization parameters.

1. Total requirement for the network throughput.
$$Min_1 = \sum e_t$$
2. Maximum requirements for the throughput of network node input ports.
$$Min_2 = \max_{n_b \in N_b} (T_i)$$
3. Maximum requirements for the throughput of network node output ports.
$$Min_3 = \max_{n_b \in N_b} (T_o)$$

As a result we get the architecture of a SpaceWire network that is based on the computing/communication workload requirements and satisfy the user-defined constraints.

3 SpaceWire network synthesis

We use decomposition of a system to subsystems. Decomposition is used for support of spatial constraints and specific timing constraints (jitter) for some tasks. It helps also to decrease algorithm complexity.

3.1 Spatial constrains

Technology constraints related to spatial placement of terminal nodes and specific of cable-laying typically are relevant for a spacecraft. Certain groups of devices should be placed locally due to their functionality (sensors, locator) or structural reasons. Typically quantity of cables that connect such group with other parts of the network is strongly constrained. This group is arranged as subnetwork includes its own set of routers.

The methodic takes into account this type of user constraints. User can specify such groups of nodes as clusters. Our tools generate subnetwork structure for each cluster independently (routers of one subnetworks are not used in others) and with constrained (required) quantity of interconnections to other network parts.

3.2 Specific timing constraints

We use network structure patterns in our methodic for specific timing constraints support.

SpaceWire networks are often used for data transmission from sensors to computer or from computer to visualization or telecommunication subsystems. Not only packet delivery time but jitter is important for these applications also. Therefore an appropriate network structure for such subsystems is

symmetric (that is important for jitter parameter) tree.

The tool generates symmetric trees of routers from the system components library that meet mass and power constraints. Adaptive routing can be used for simultaneous throughput utilization of some links, which directly connects two devices (e.g. in fat trees).

Such subsystems can include up to 90% of terminal nodes. Therefore this approach also allows to essentially decreasing the computation complexity of the integrally algorithm of network structure generation. (Basic variant of network structure generation algorithm for arbitrary network is NP-hard.)

At a lower layer of the tree our tool could generate daisy chains of nodes (if nodes supports this functionality), to decrease hardware cost of the network.

The tool automatically selects subgraphs in a logical interconnection graph, for which tree based subnetworks could be generated.

An example of tree subnetwork generation is represented on fig 1. The logical graph structure is represented on left part of the figure. The logical graph structure includes two subsystems. They are marked “Tree1” and “tree2”.

For every subsystem separate tree network is generated. After that these subnetworks are connected together. The result network interconnection structure is represented in middle part of the figure. The network interconnection structure when daisy chain connections for terminal nodes are possible.

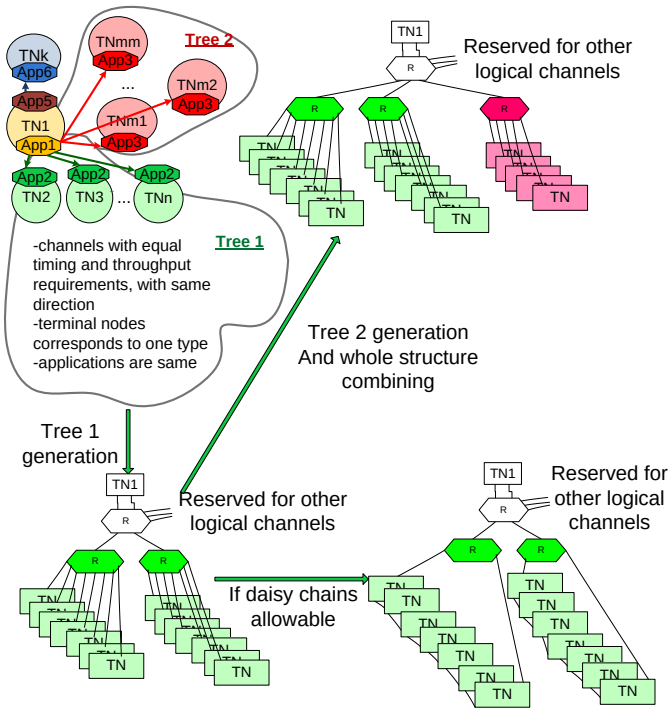

Fig..1 Example of tree subnetwork generation

Others typical for a SpaceWire networks structures are used for distributed computing. In the tasks graph such structures are usually represented by subgraphs with peer to peer connections between tasks. Typical requirement is equal data transmission time between all components of a distributed computing platform.

A designer should select subgraphs that correspond to distributed computing in a logical interconnections graph because rules of such subgraphs detection strongly depends on the tasks that are processed in system.

A good structure for such network fragments corresponds to bipartite graphs is Banyan network [7,8,9]. Banyan networks provide path with equal length between all nodes from one set of nodes from a bipartite set to another set.

In our methodic Irregular Banyan subnetworks are built of routers. The Banyan subnetwork generation algorithm can use the adaptive routing to utilize total throughput of some links that directly connect a pair of devices (nodes, routers) and takes into account requirement of connections with other parts of the SpaceWire network.

Subnetworks for logical interconnections subgraphs with peer-to-peer interconnections are generated as Banyan networks also.

A Banyan network in this case should connect not only nodes from different sets but nodes from one set with same timing parameters of interconnections also. We use coupled Banyan networks in what in the left half of the network we append interconnections mirrored interconnections of the right half; in the right half we append interconnections mirrored interconnections of the left half.

Example of such a network is represented in Fig.2. Black lines correspond to interconnections between nodes from different sets, gray lines correspond to interconnections between nodes from one set.

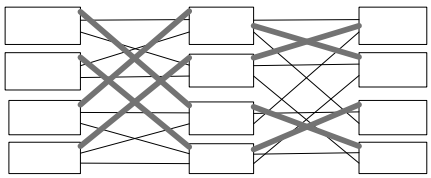

Fig.2 Example of doubled banyan network

4 Synthesis of Logical Network structure

Synthesis of a logical network structure includes mapping of logical channels (created at the architecture synthesis stage) to physical paths (in the basic variant of network structure) and generation of configuration settings for all terminal nodes and routers in the network structure.

In the developed algorithm, which maps logical channels to physical paths, logical channels could be mapped not only to shortest paths, by to others paths that correspond to throughput and timing constraints. Usage of the adaptive routing for throughput is also considered.

Mapping of logical channels to physical paths starting with check of logical channels and links characteristics. Further physical path are searched for one logical channel from list successively. Logical channels can be sorted according to priority of logical channels or user preferences. When searching for physical paths information about logical channel characteristics (for example throughput) is taken into account. This procedure is marked as successful, when physical paths are created for all logical channels. Several physical paths can match logical channel characteristics thus a few solution can be created. Conclusive physical paths for all logical channels are selected at final step. The principles of selection are different. For example, shortest physical paths can be selected. User specifies this criterion at the beginning.

Further generation of configuration parameters for terminal nodes and routers (routing tables' content, adaptive routing mode, and link transmission rate) is performed.

Link transmission rate is configured correspondingly to required throughput and packet transmission time. Adaptive routing could be configured for pairs of nodes or routers that are directly connected via some links (for throughput).

Logical addresses of the terminal nodes should be defined before the routing table content generation. Our tools assign a logical address to every application (task) in a terminal node.

If the quantity of addresses is more than 224, then the regional addressing is used. Regions (or groups of regions) correspond to subgraphs that are extracted in the logical interconnection graph structure.

Routing tables' content is generated after logical and regional address assignment, in correspondence to mapping of logical channels to physical paths.

Example of network configuration is represented on fig. 3. Network consists of 7 terminal nodes and 3 routers. Paths of logical channel are displayed by colored lines with arrows. Links are shown by black lines. TX_speed configuration, Routing tables' content and information about adaptive routing are presented on this figure.

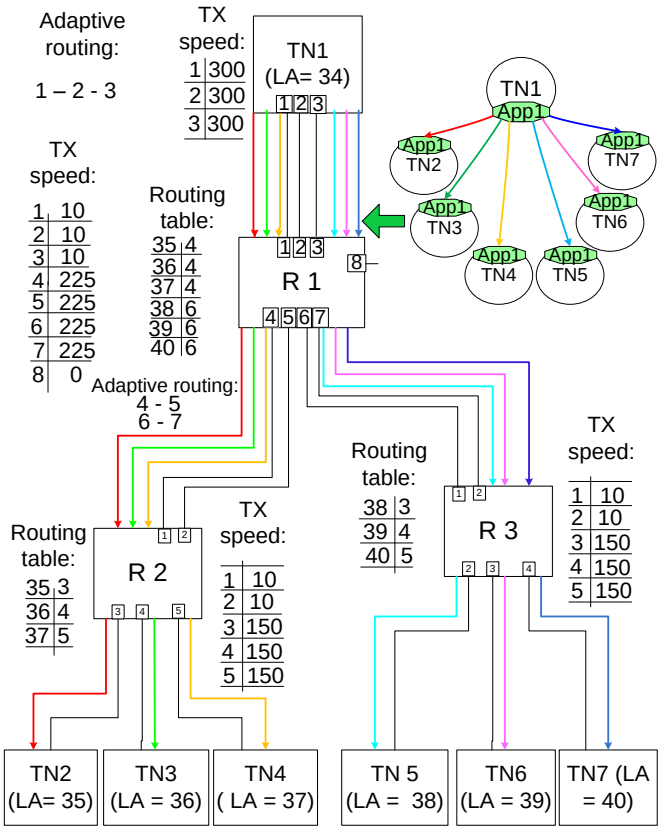

Fig.3 Example of network configuration

5 Fault tolerance support

Hardware redundancy is used for providing fault tolerance.

Fault tolerance support is realized by to hardware replication (routers and links replication). We suggest two types of redundancy policy: path replication based and dynamic path reconfiguration.

5.1 Path replication based redundancy

If tolerance to N-1 faults is need, the whole SpaceWire network structure should include N independent paths between source and destination terminal nodes. The whole network structure (Fig.4) includes N copies of the basic structure (configurations of corresponding routers in all replicas are identical). Every terminal node is connected to all replicas of the basic network structure. Source terminal nodes should send copies of every packet to all replicas of the basic structure. Destination terminal nodes should correctly interpret data flows with proper and faulty copies of received packets.

This redundancy policy is recommended for systems with strong requirements to packet delivery reliability and real time constraints.

Hardware cost of buffering scheme in terminal nodes is essential for this redundancy policy.

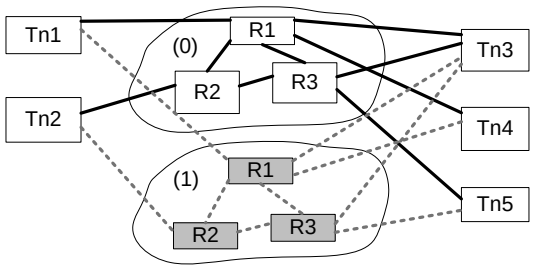

Fig.4 Example of a path replication based network structure (N=2)

If fault tolerance is required, at the first stage our tools generate a basic network structure and logical configuration for it and next this network structure is expanded for fault tolerance.

User should reserve resources (mass, power, number of device's used ports that can be utilized for a basic network structure, should be decreased correspondingly to quantity of faults and selected fault tolerance policy) before a basic network generation. In the basic network should be used not more than 1/N allowable mass and not more than 1/N of every terminal node port's quantity for tolerance to N-1 faults.

For practical reasons it is typical to apply FT-requirements to some fragments of the network only, to some of its subnetworks, clusters. Thus the strategy of network redundancy by path replication is applied to these individual parts of the network.

5.2 Dynamic path reconfiguration redundancy

Further, dynamic path reconfiguration redundancy could be used for a N-1 faults tolerant network that includes N replicas of a basic network (Fig.5). All routers Ri(0) and Rj(0) in the basic network (marked by "0") that are directly connected, have connections with routers Rj(k) and Ri(k) for all network replicas (represented by gray dotted lines in the figure).

Adaptive routing configuration for direct interconnections of devices by some links is identical in the basic network and all the replicas.

For dynamic path reconfiguration additional adaptive routing configuration is generated for interconnections between network replicas in whole network structure.

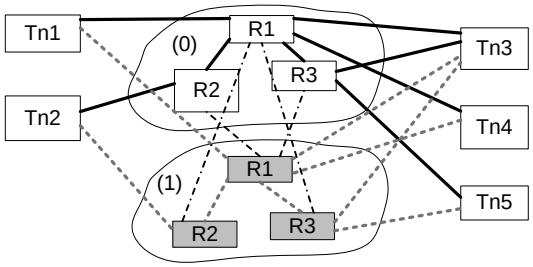

Fig.5 Example of dynamic path reconfiguration based network (N=2)

In this case a terminal node sends to the network only one copy of a packet. It sends packet to any link that corresponds to this packet's path and is currently in the work state. Then every next transit router send packet to any link that corresponds to this packet's path and is ready. If a fault occurs in a router or a link when a packet transmitted, this packet would be lost or corrupted and destination node would not receive correct copy of it. Also fault can impact to transmission time of others packets in network.

However in this case designer doesn't need additional hardware for packet buffering in terminal nodes. Therefore this fault tolerance policy is recommended for networks without guaranteed delivery packets, without strict real time requirements and when packets buffering in terminal nodes is impossible.

When a designer plans to use this fault tolerance policy, he should reserve resource of network equipment mass and resource of terminal nodes and routers port's quantity before a basic network generation. In the basic network should be used not more than 1/N allowable mass and not more than 1/N of every router and terminal node port's quantity for tolerance to N-1 faults.

6 Conclusion

The paper describes the methodology and toolset for SpaceWire network design. It provides the design space exploration mechanism for synthesis of the large SpaceWire networks. The design includes the set of terminal nodes, communication structure, switches and links to meet the computation requirements and user-defined constraints. The high level of automation allows making changes in requirements and reconfiguration of SpaceWire network rapid and easy.

Acknowledgment

The research leading to these results has received funding from the Ministry of Education and Science of the Russian Federation under grant agreement n 14.578.21.0022.

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to algorithms, Second Edition, Massachusetts Institute of Technology, 2009.

[2] Hazewinkel, M. (2001), "Steiner tree problem", in Hazewinkel, Michiel, Encyclopedia of Mathematics, Springer, ISBN978-1-55608-010-4

[3] Algebraic Graph Theory" by Chris Godsiland Gordon Royle, published by Springer-Verlag, 2001, 439 pp

[4] The Steiner three problem: a tour through graphs, algorithms, and complexity Hans Jürgen Prömel, Angelika Steger. Amer Mathematical Society, 2002 241 pp.

[5] Wu, Bang Ye; Chao, Kun-Mao (2004). Spanning Trees and Optimization Problems. CRC Press. ISBN1-58488-436-3

[6] A Walk Through Combinatorics: An Introduction to Enumeration And Graph Theory. Miklós Bóna. World Scientific, 2006 – 469 pp.

[7] Principles and Practices of Interconnection Networks William James Dally, Morgan Kaufmann Publishers is an imprint of Elsevier 2004 581 pp

[8] Computer Networks V.S.Bagad, I.A. Dhotre, Technical Publications, 2009 г. – 512 pp.

[9] Data And Computer Communications. William Stallings. Pearson Education, 2007.852

7 Contact Author Email Address

mailto: nadezhda.matveeva@guap.ru

Copyright Statement