BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY
DEPARTMENT OF CONTROL ENGINEERING AND INFORMATION TECHNOLOGY

György Pilászy

New algorithms for extennding the high level synthesis methodology to designing and optimizing distributed systems

PhD thesis summary

Supervisor: Dr. Péter Arató
Advisor: Dr. Géza Móczár

The digital version is available in the following URL:

http://directory.iit.bme.hu/~gpilaszy/phd/

Budapest, 2013

Introduction

Due to the permanent technological development, a single chip nowadays may consists of even more hundreds of millions transistors. Therefore, very high-performance VLSI components (FPGA-s, processors, etc.) became available on extraordinary low price. Among such components there are also the microcontrollers and the diverse signal processing units with memory and suitable input/output facilities. These complex signal processing units are able to solve increasingly complex tasks independently as components of a distributed system.

For designing and optimizing distributed systems containing the above complex components, the existing more or less systematic system level synthesis (HLS) methods, like the high level synthesis algorthms are not applicable without changes and extensions. These methods generally starts from a datafow graph obtained by decomposing (partitioning) the task specification described by a high level programming language (e.g. C) [7, 30]. The existing design tools attempt to optimize directly from the dataflow graph supposing relatively simple execution unit as components. These existing tools are not able to handle complex processing units specified by the design procedure.

The dissertation aims to extend the methodology of the high level synthesis for making it applicable in the systematic design and optimization of distributed (multiprocessing) systems.

Specifically:

- To elaborate a method for handling the transfer times of the communication buses between the complex processing units.
- To examine the effect of incrementing the latency in pipeline systems.
- To elaborate a method for the impact assesment, whether a calculated latecy increment could decrease the cost at preserving the throughput (restart time).
- To develop an algorithm for fulfilling predefined separation and fusion conditions in the sysnthesis procedure.

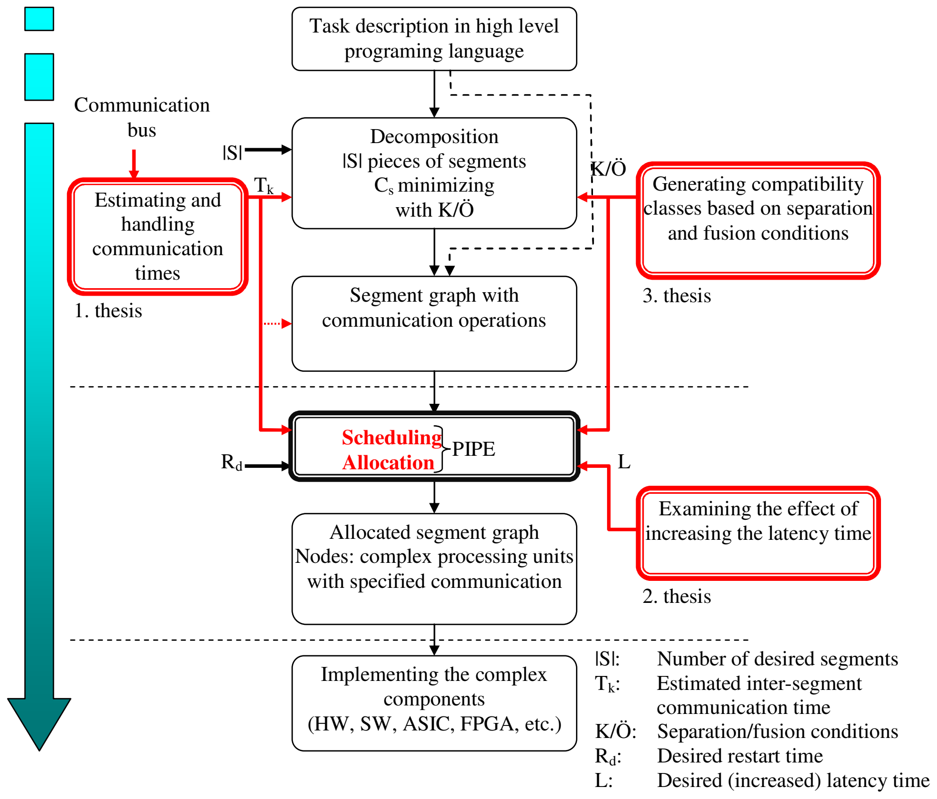

Task description in high level programing language
Communication bus
|S|
Decomposition |S| pieces of segments C_s minimizing with K/Ö
K/Ö
Generating compatibility classes based on separation and fusion conditions
T_k
Estimating and handling communication times
1. thesis
3. thesis
Segment graph with communication operations
Scheduling
Allocation
PIPE
R_d
L
Examining the effect of increasing the latency time
Allocated segment graph Nodes: complex processing units with specified communication
2. thesis
Implementing the complex components (HW, SW, ASIC, FPGA, etc.)
|S|: Number of desired segments
T_k: Estimated inter-segment communication time
K/Ö: Separation/fusion conditions
R_d: Desired restart time
L: Desired (increased) latency time

Figure 1. Extension of the HLS methodology to systematic design of distributed systems

The design procedure of distributed systems -in the above sense- is summasized in Figure 1. The extensions and the new algorithms are illustrated in thick frames.

The results of the dissertation

1. Communication time estimation

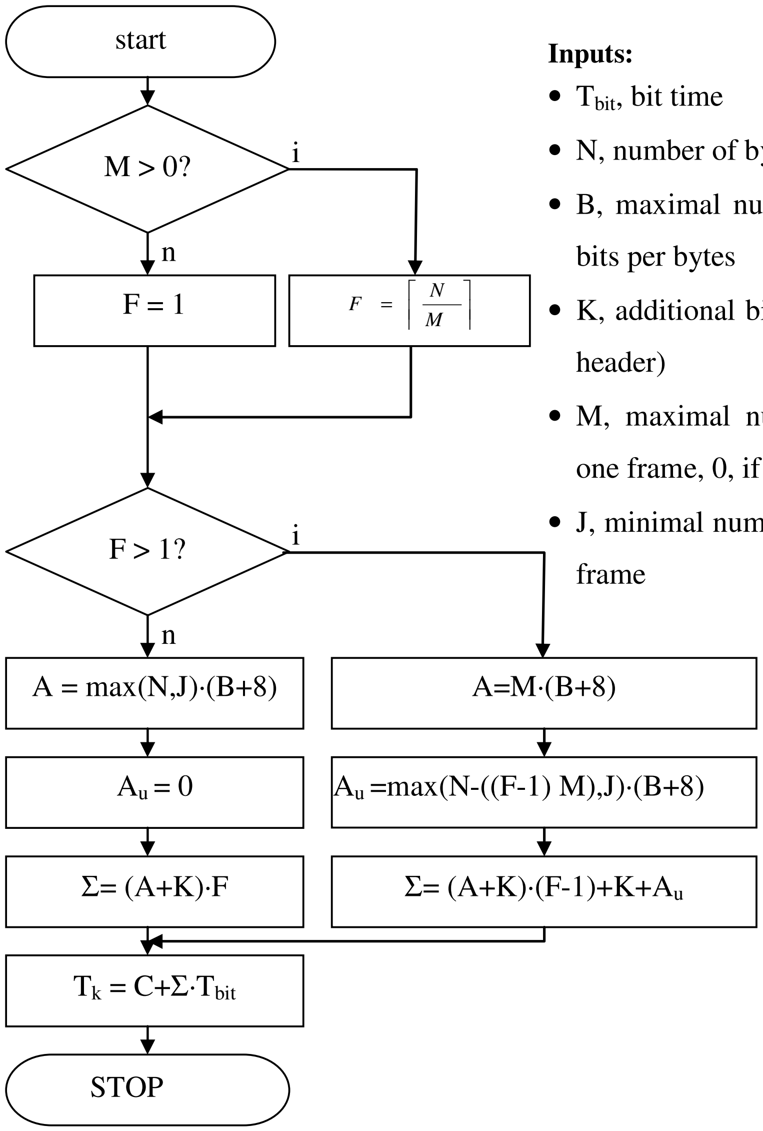
The most existing HLS tools [3, 25] do not assume any time demand for the communication between the operations in the initial dataflow graph or between the complex processing units specified by the decomposition and allocation steps. In [6, 31, 32] there are arguments for the importance of handling the communication properly, but the communication is characterized only by the number of bits to be transferred and by some basic properties without calculating the time demand [6]. Such an approach is acceptable inside an FPGA, but between the separate complex processing units (e.g. microcontrollers, microprocessors with integrated communication channels), the proper time demand estimation for the communication cannot be avoided just at the beginning of the design procedure. [5]

The methodology presented in this dissertation handles the communication as an additional elementary operation. In this case, the communication time demand is represented by the execution time of this additional operation. The exact communication time is not always known in advance (e. g. because of data-dependent bit insertion), therefore a realistic estimation is crucial. Underestimation yields function especially in pipeline mode. Overestimation slows down the resulting system unnecessarily. A proper upper limit in the estimation may help in avoiding such problems.

By analysing the bus systems as examples in the dissertation, the following main properties can be observed:

- The smallest data unit to be transferred is one byte in most interface systems.
- The data bytes are often provided with additional bits (e.g.: start, stop, ack).
- To the communication belong both an additional bitfield of fix length and an additional bitfield of variable length in each frame (package).
- The communication time-demand can properly be estimated as the integer multiple of the bit transfer time.

Figure 2 shows the flow chart of the time estimation algorithm.



Inputs:

- T_{bit}, bit time
- N, number of bytes
- B, maximal number of additional bits per bytes
- K, additional bits per frames (i.e.: header)
- M, maximal number of bytes in one frame, 0, if no restriction
- J, minimal number of bytes in one frame

Figure 2 Communication time estimation

The parameters for the four bus systems analysed in the dissertation for some further bus sytems (not detailed here) are summarized in Table 1. The descriptions of the latter communication interfaces can be found in [S4, 27, 28, 29, 30].

Based on the algorithm, Fig. 3 shows the communication times of the bus systems analysed in details by assuming a data transfer speed 1Mbit/s and a data transfer 1-32 bytes. It can be observed for the CAN bus that the limited 8 byte frame increases the communication overhead, since further frames are required for transferring longer data.

Interface	T_{bit}	C [µs]	B [Bit]	K [Bit]	M [Byte]	J [Byte]
SPI	50ns min.	0	0	1	0	0
I^2C, (100kHz-1MHz) 7bits address	1-10µs	0	1	10	0	0
I^2C, (100kHz-1MHz) 10 bits address	1-10µs	0	1	18	0	0
I^2C, High speed (3.4MHz), 7 bits address	295ns	25	1	10	0	0
I^2C, Ultra high speed (5MHz),7 bits address	200ns	0	1	10	0	0
UART (1 stop, 0 pariy)	100ns min.	0	2	0	0	0
UART (1 stop, 1 parity)	100ns min.	0	3	0	0	0
CAN 2.0A	1µs min.	0	2	56	8	0
CAN 2.0B	1µs min.	0	2	81	8	0
BME-PS CAN plus	1-10µs	0	2	113	4	4
Ethernet 802.3. (10Mbps)	100ns	0	0	26	1500	46
USB low-speed (1.5Mbps), control	666ns	0	1	504	8	0
USB low-speed (1.5Mbps), interrupt	666ns	0	1	152	8	0
USB full-speed (12Mbps), control	83.3ns	0	1	360	64	0
USB full-speed (12Mbps), Interrupt	83.3ns	0	1	104	64	0
USB full-speed (12Mbps), Isochronous	83.3ns	0	1	72	1023	0
USB full-speed (12Mbps), Bulk	83.3ns	0	1	104	64	0
USB high-speed (480Mbps), control	2.083ns	0	1	1384	64	0
USB high-speed (480Mbps), Interrupt	2.083ns	0	1	124	1024	0
USB high-speed (480Mbps), Isochronous	2.083ns	0	1	112	1024	0
USB high-speed (480Mbps), Bulk	2.083ns	0	1	440	512	0
SATA (1.5GB/s), parancskeret	666.6ps	0	2	96	64	64
Hyper Transport (2bit,200MHz)	2.5ns	0	0	64	64	4
Hyper Transport (32bit,200MHz)	156.25ps	0	0	64	64	4
PCI-Express x1 (2.5Gb/s)	400ps	0	2	144	4096	0
PCI-Express x2 (2.5Gb/s)	200ps	0	2	288	8192	0
PCI-Express x4 (2.5Gb/s)	100ps	0	2	1152	16384	0
PCI-Express x8 (2.5Gb/s)	50ps	0	2	2304	32768	0
PCI-Express x16 (2.5Gb/s)	25ps	0	2	4608	65536	0
PCI-Express x32 (2.5Gb/s)	12.5ps	0	2	9216	131072	0

Table 1 Parameters for estimation algorithm

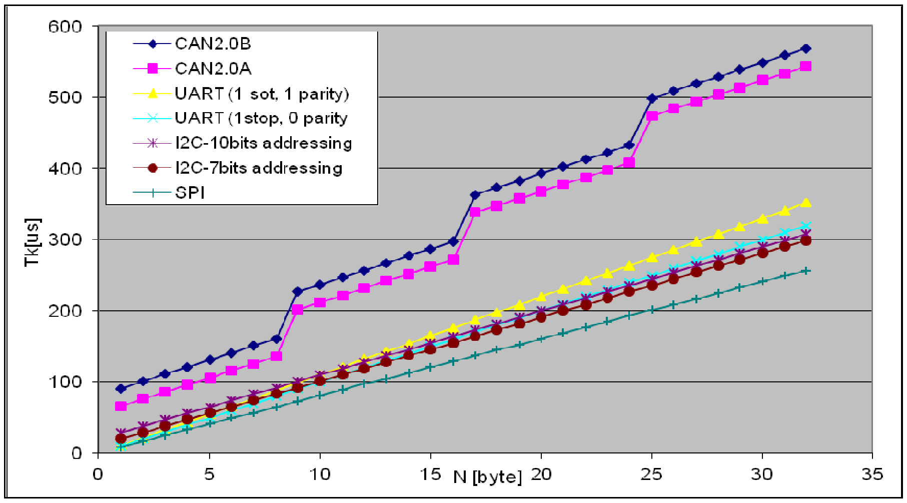


Figure 3 Communication time for 1-32 bytes with 1us T_{bit}

Applying the results in extending the HLS algorithms

The HLS tools usually represent the task to be solved by an initial dataflow graph (EOG) called often elementary operation graph (EOG). In Figure 4 a simple example is shown for inserting the additional communication operation (e_3) between the elementary operations e_1 and e_2. For illustrating, the EOG of HLS tool PIPE is applied.

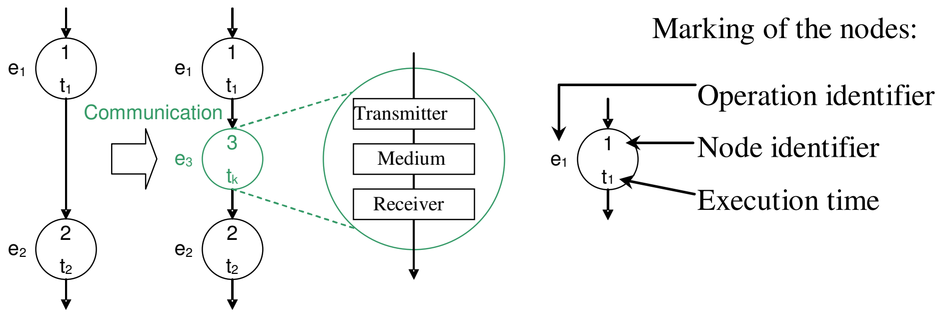


Figure 4 Representation of communication in EOG

The execution (latency) time of operation e_3 is t_k. Since the communication time T_k depends on the bit transfer speed, therefore it has to be transformed into the timing system of the HLS tool. In tool PIPE, the basic time unit is the clock period T, thus:

$$t_k = \frac{T_k}{T}$$

According to one of the solutions proposed in the dissertation, the communication operation should be inserted into each data connection of the initial EOG at the beginning of the design. After the decomposition, when the segments become known, t_k=0 values have to be set inside the segments. This situation is illustrated in Figure 5.

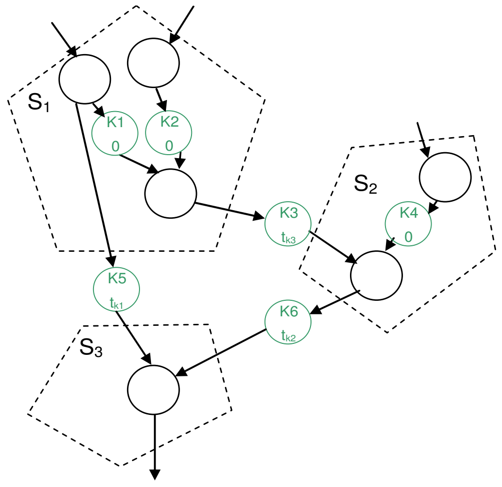
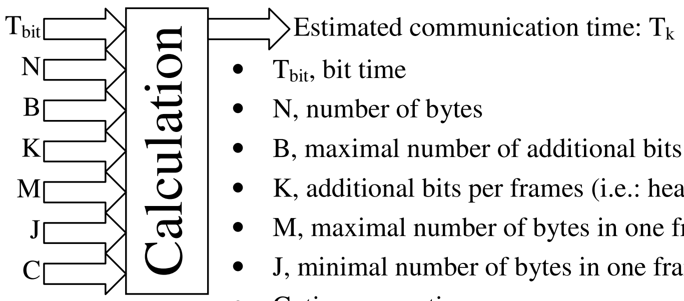


Figure 5 Illustration of interpreting the communication operations inside the segments and between the segments

1. Thesis

- **The concept of elementary operations can be extended to operations with arbitrary complexity by specifying them at the beginning of the design procedure or by producing them during the decomposition or allocation. Between such complex components (microprocessors, microprocessors, signal processing units, etc.), the communication time has to be taken into consideration.**
- **I elaborated a method for handling the communication as additional operations between the complex processing units. The execution time of such additional operation is the calculable or estimable communication time.**
- **I developed an algorithm for calculating and estimating the communication time. The algorithm is illustrated for some often used bus systems.**



- T_{bit}, bit time
- N, number of bytes
- B, maximal number of additional bits per bytes
- K, additional bits per frames (i.e.: header)
- M, maximal number of bytes in one frame, 0, if no restriction
- J, minimal number of bytes in one frame
- C, time correction

- I have shown on an example that the traditional HLS method may result in structure which could not be implemented by applying the usual embedded communication channels.
- I have shown that the extended HLS (PIPE) method yields an implementable solution by applying the algorithm I have elaborated.
- The elaboration of the method and the algorithm are my own results
- The publications containing the results of this thesis: [S2], [S6], [S8] and [S9].

2. The effect of increasing the latency time in high level synthesis

In pipeline systems the HLS tools attempts to optimize the restart (initialisation) time (R) and the latency (L) is an output parameter obtained at the end of the calculations [3]. It is not examined, whether the intentional increasing of the latency how could influence the complexity and the cost in pipeline systems. This is the aim of the following exanination by applying the HLS tool PIPE, but the conclusions remain valid for other tools as well. The cost reduction can be expected if originally overlapped operations become not overlapping due to longer latency by proper scheduling. The operations not overlapping any more may be implemented in the same execution unit by the allocation.

Necessary (but not satisfactory) rules for cost reduction:

1. Obviously, only such execution units are worthy to be examined which occur in more copies in the allocated EOG.
2. It is beneficial to specify the trivially lowest cost. Having achieved this cost value, further cost reduction cannot be expected. For example: if an EOG consists of four different operations which cannot be implemented by common execution unit, then at least four execution units are required for the solution.
3. The minimal required copies (number of pieces) of the different execution units can be calculated for a given restart time. Based on this result, the theoretically achievable minimal cost can be calculated. This cost value should be considered as a stop condition for examining the effect of increasing the latency.
4. In the case of identical operations started at the same time, the latency should be incremented at least by the execution time of the operation, because overlapping operations cannot be implemented by common executing unit.
5. Overlapping operations can be scheduled after each other by a latency incrementation at least the overlapping value, if there is no other exclusion condition.
6. For using the complex execution unit M_i k-times during a restarting period, the sum of the execution times (t_i) of the k pieces of elementary operations implemented by M_i must not be greater than the restart time: $\sum_k t_k \leq R$

   If the execution times of the elementary operations allocated in M_i are identical, then:

   $k \cdot t_k \leq R$

In the dissertation, more example is shown for the impact assessment of increasing the latency time. Figure 6 shows for instance the EOG of task F2.

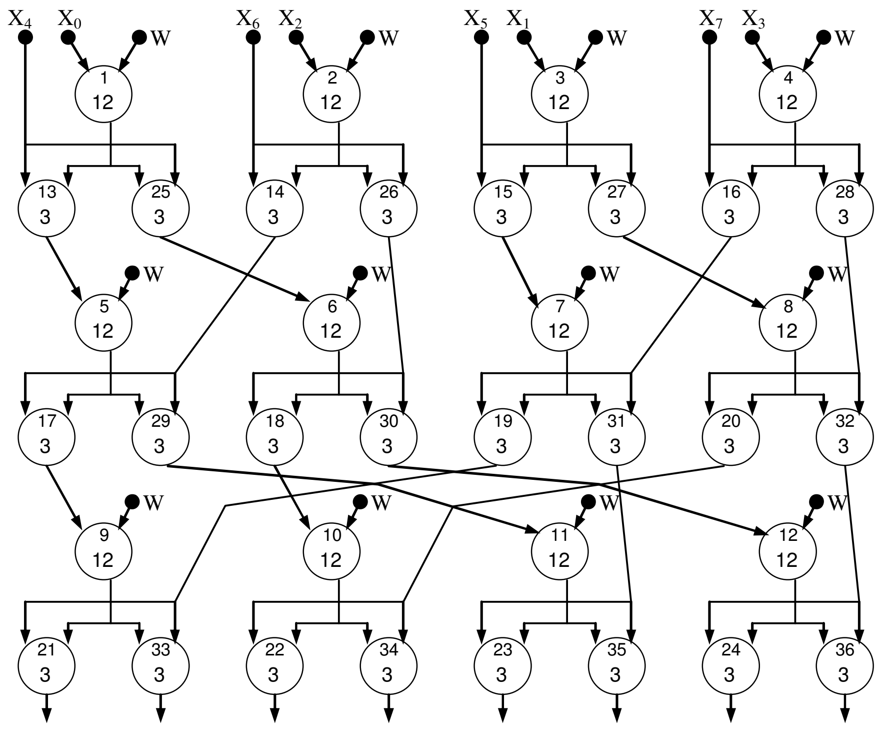


Figure 6 The F2 EOG

The properties of the three execution units assumed in the EOG are summarized in Table 2.

execution unit	Elementary operations	t_i	pieces in EOG
P_1	$e_1 \ldots e_{12}$	12	12
P_2	$e_{13} \ldots e_{24}$	3	12
P_3	$e_{25} \ldots e_{36}$	3	12

Table 2 The properties of the execution units

The comparison of the various solutions has been made according to a very simple cost calculation that is the product of the number of the applied execution units and their execution times Fig. 7 and Fig. 8. The lowermost curves correspond to the theoretical cost minimum at a given restart time. It can be observed that this cost is approached closer at longer latency values.

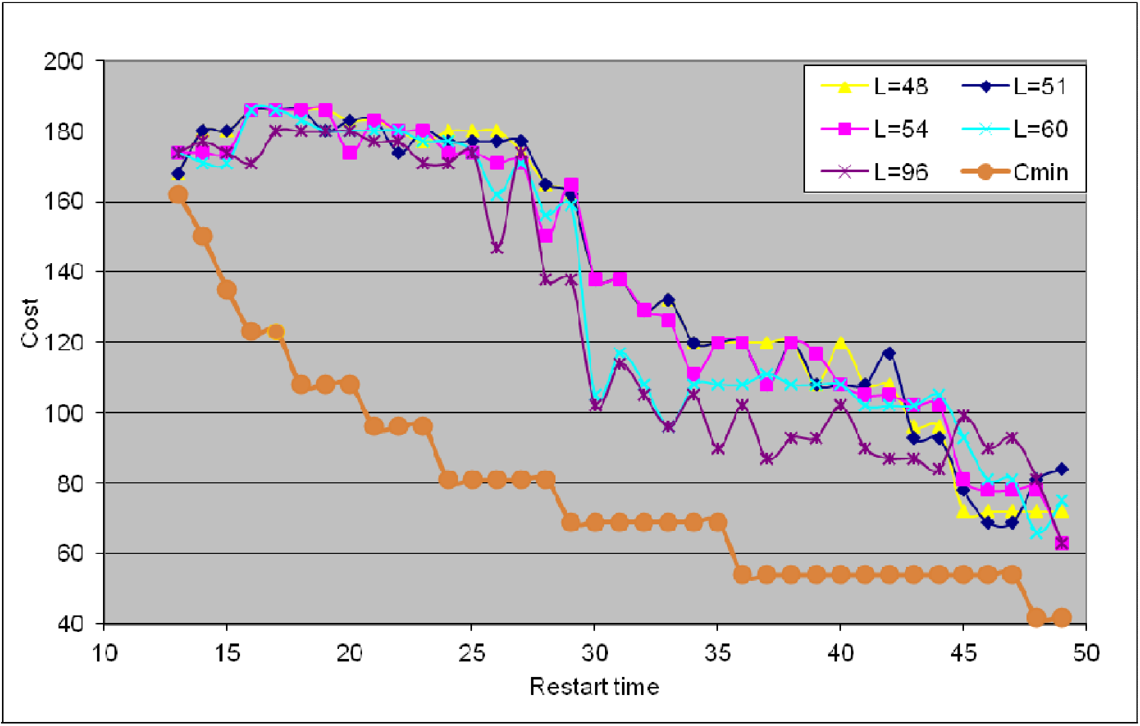


Figure 7 Cost functions of F2 in case of three different execution units

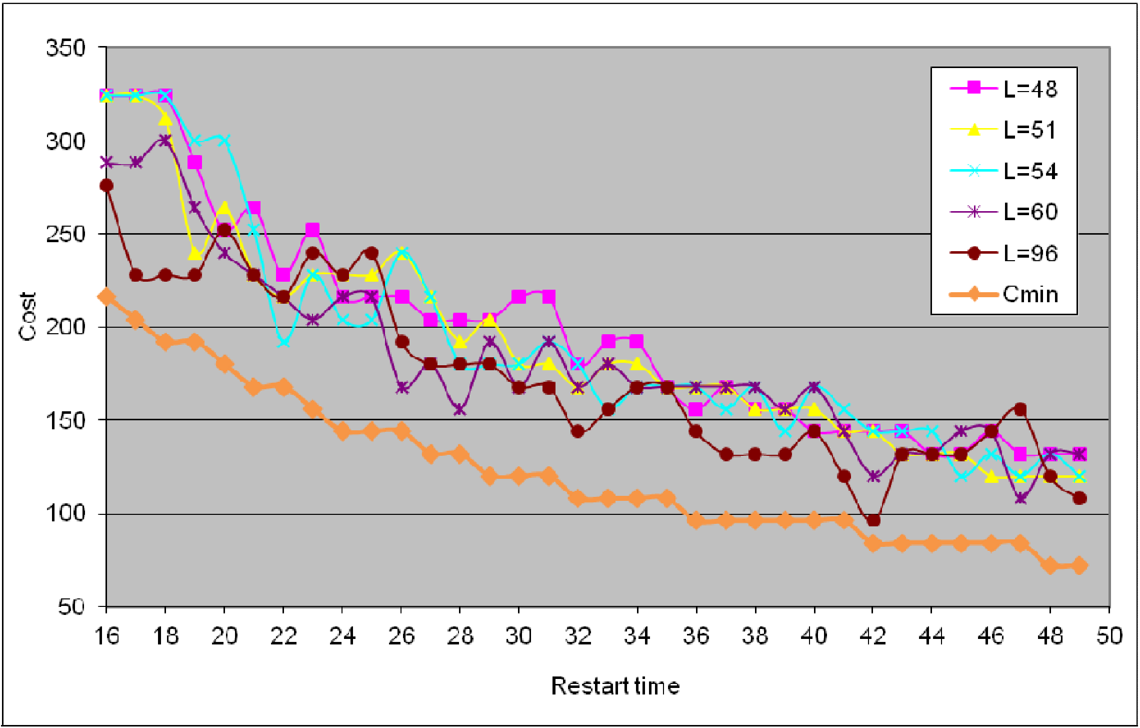


Figure 8 Cost functions of F2 in case of complex execution units

The algorithm proposed for the impact assessment of the latency incrementation

The parameters and notations for the algorithm are defined in the dissertation, only the resulted procedure are presented here.

1. Run PIPE on the initial EOG for the given R. Based on the result, L_{min} is obtained.
2. Determine the minimal number of pieces (N_{iMIN}) of each execution unit P_i; $N_{iMIN} = \sum_k \left\lceil \frac{n_k \cdot t_k}{R} \right\rceil$; k refers for the operations which are executable by P_i .
3. Calculate C_{min} .
4. Generate partition Π_M based on set M. Select the blocks ($m_{i)}$ of the partitition for which $N_i > N_{iMIN}$ and $T_i \leq R/2$ stand.
5. Determine a latency increment value Δ as the execution time T_i belonging to m_i having the greatest N_i (In case of more greatest N_i-s, then select from them according to the smallest T_i).
6. Increment the latency by the selected Δ.
7. Rerun PIPE
8. Calculate the cost C. Check the stop conditions, if not fulfilled, repeat from step 4.
9. Stop if either from the following conditions are fulfilled:
   - if the expected cost (C_d) given in advance achieved or
   -if the maximal latency (L_{max}) given in advance is achieved or
   -if the minimal cost value (C_{min}) is achieved.

The simplified flow chart of the algorithm is shown in Figure 9.

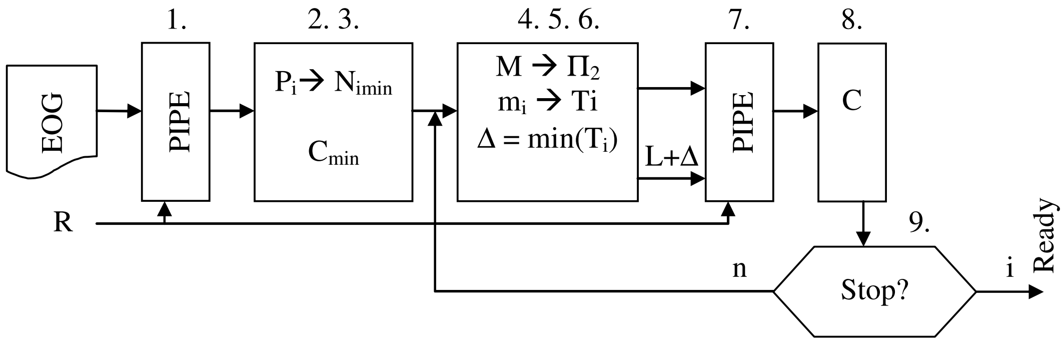


Figure 9 The flow chart of the algorithm for calculating the latency incrementation

The aim of the above algorithm is to aviod unnecessary reruns of PIPE. The rerun is started only if the latency increment is expected to cause cost reduction. Obviously, it does not make sense to increase the latency extremly, because the system would slow down without reason.

2. Thesis

In pipeline systems, the HLS tools attempts to optimize the restart time (R) and the latency (L) is an output parameter obtained at the end of the calculations. It is not analysed, whether the intentional increasing of the latency how could influence the complexity and cost in pipeline systems.

- ➢ **In the case of pipeline systems, a longer latency may cause cost reduction without slowing down the system, because the pipeline throughput are not affected significantly.**
- ➢ **I have elaborated a method and an algorithm for the impact assessment of incrementing the latency in the design procedure of distributed pipeline systems.**

- I have evalueted the results by simulation.
- I have developed a method for calculating beneficial increment values in increasing the latency.
- The method allows the jointly modification of the restart time and the latency in order to approach the optimal cost.
- Based on my proposal, the HLS tool PIPE has been modified in order become able to receive the latency as an input parameter.
- The method and the algorithm summarized in this thesis are my own results.
- The publication containing the results of this thesis: [S3].

3. New allocation method for predefined separations and fusions

In distributed systems (e.g. in the automatization of industrial systems), the improper task partitioning may cause difficulties. Although the whole system perform its specification, if each component (segment) functioning properly, but a malfunction of a component may prohibit the proper operation of such other components, which are affected only by the improper task partioning and not by funtional reason. Thus, the toublshooting may become extremly difficult and time-consuming. Likewise, the improper partitioning may implement closely related task functions into different components (e.g. placed far from each other). Such a distribution of a function makes difficult the functional testing of the sytem, because all affected components should operate properly at the same time.

To aviod the above difficulties, predefined separations and fusions have to be ensured by the decompostion and allocation. The existing HLS tools are not prepared for fulfilling such boundary conditions.

The above problems are comparable with some boundary conditions in hardware/software codesign [22, 23, 24]. Namely, the task distribution must predefintely implement some functions into hardware components (e.g. ventils, relays, sensors), and some others into software (e.g. programs for PLC or microcontrollers). There are also such task functions, which can be implemented either hardware or software. Besides, prescriptions in standards [11] (e.g. safety-critical rules) may also to be considered as predefinit boundary conditions.

Figure 10 illustrates the main steps of HLS [3, 24] affected by the separation and fusion conditions.

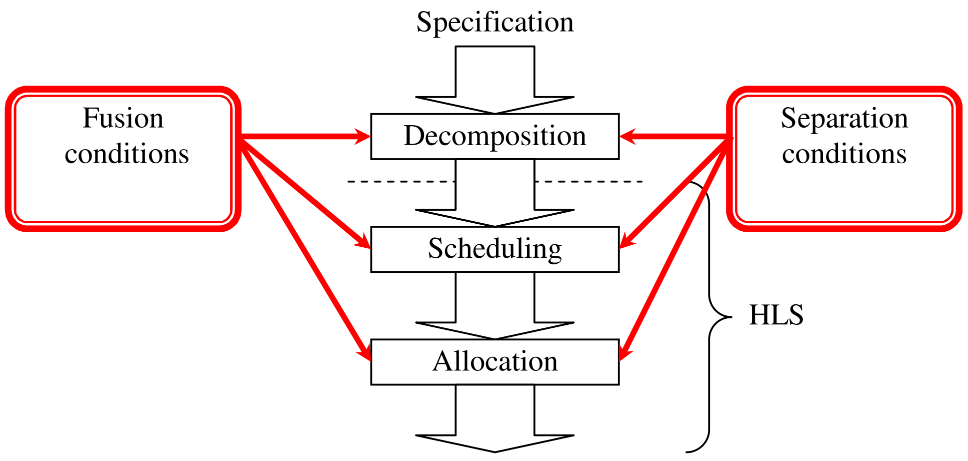


Figure 10 The HLS steps sffected by fusion and separation conditions

Further on, the modification algorithm is detailed affecting the allocation step. Ti can be proven that the pairwise fusion condition of the elementary operations is a compatibility relation [12, 13]. Based on the maximal compatibility classes, a proper disjoint closed cover should be generated by the allocation.

The proposed new algorithm for the allocation

The main steps of the algorithm detailed in the dissertation are as follows:

1. Based on the predefined boundary conditions, generate the compatibility half matrix (CHM) containing the prescribed fusions, separations and conditional fusions.
2. If CHM contains conditional fusions, then ensure the closure of the condition chain by introducing proper separations (if necessary).
3. Generate the maximal compatibility classes.
4. Check the disjointness of the classes.
5. If the classes are not disjoint, then check if there are any classes to be neglected without hurting the closure. Such classes should be left out. If this neglecting hurts the closure, check the effect of neglecting only a subclass or rather separating an other class. This step is task-dependent and requires trials.
6. Ensuring the disjointness and the closure of classes may result in increasing number of classes.
7. If extremely great number of not disjoint classes are obtained, then the inequality $M \leq P \leq \min(n,k)$ [13] can provide information on the „goodness” of the actual results in order to avoid further trials.

P:	the number of classes in the actual result,
n:	the number of nodes in the input graph for the allocation,
k:	the number of maximal compatibility classes generated from the initial CHM,
M:	the number of maximal compatibility classes representing the simplest closed cover.

8. Each class of the result represents those operations from the input graph of the allocation, which should be allocated in a common execution unit (component).

The flow chart of the algorithm is illustrated in Figure 11.

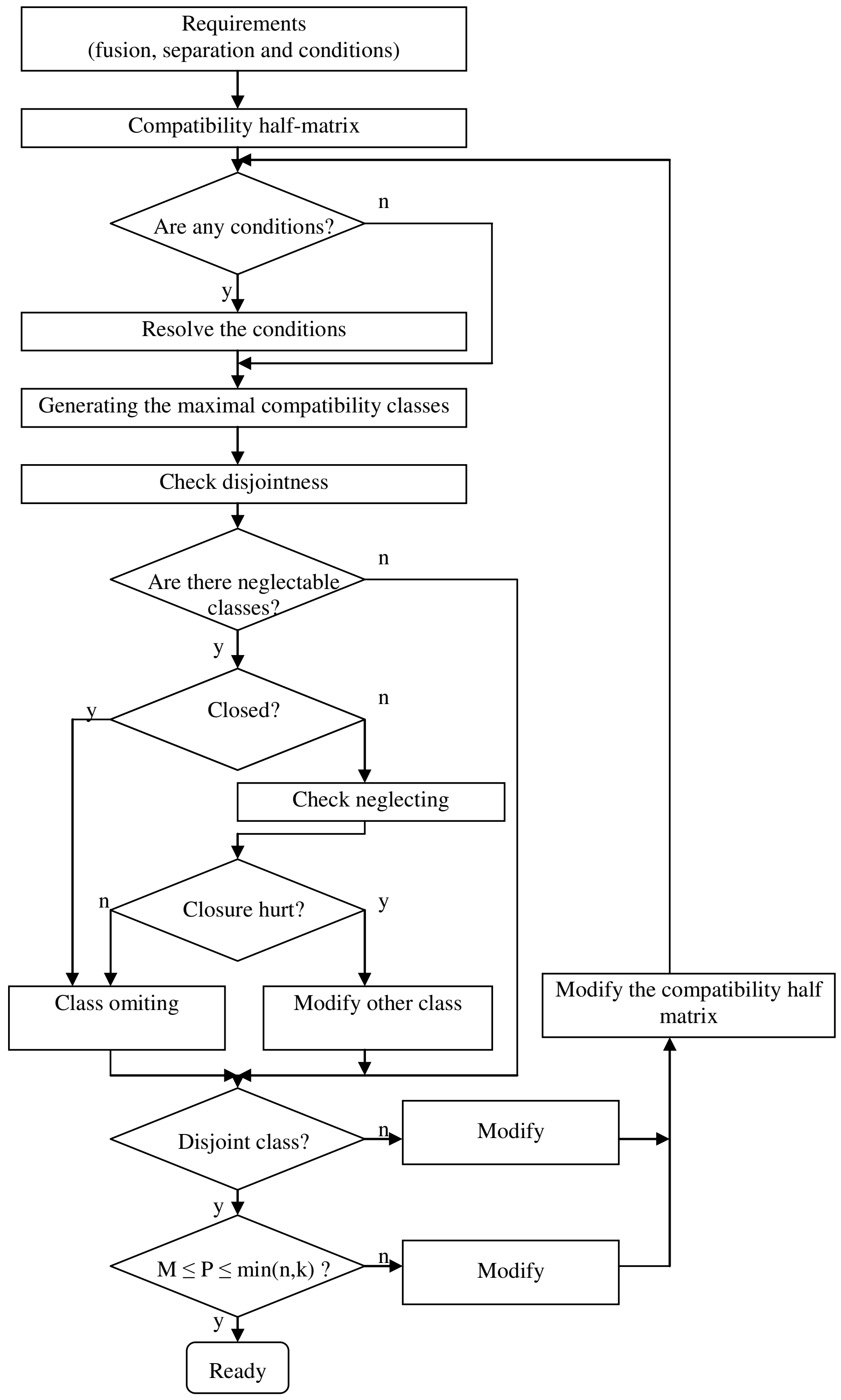


Figure 11 The flowchart of the allocation algorithm

3. Thesis

- ➢ **In the high level synthesis of complex distributed systems, operating, maintanance, standard and safety-critical prescriptions have to be taken into consideration beside the task functionality.**
- ➢ **I have elaborated a new algorithm for the allocation in high level synthesis, which – unlike the existing tools- can handle and consider predefined boundary conditions for fusions, conditional fusions and separations of operations.**

- The algorthm can be applied in the systematic design of safety-critical complex industrial distributed systems.
- The algorithm can be applied in the allocation of communication operations (detailed in Thesis 1) for influencing according to the resouces.
- Based on the algorithm, developing new scheduling and allocation procedures have been started.
- The method and the algorithm summarized in this thesis are my own results.
- The publications containing the results of this thesis: [S9], [S10].

Acknowledgement

I am very grateful to my supervisor, Dr. Péter Arató and to my advisor, Dr. Géza Móczár for their help in my research and developing work.

The support of the Hungarian Scientific Fund (OTKA K72611) and New Hungary Development Plan (Project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0002) IKT-P5-T3 are gratefully acknowledged. This work is also belonging to the scientific program of the "Development of quality-oriented and harmonized R+D+I strategy and functional model at BME" project.

Publications

Referred journal papers in English

S1. Komáromi Zoltán, Pilászy György, Móczár Géza, "Hardware-software co-design in control engineering based on MATLAB environment", ISAST TRANSACTIONS ON COMPUTERS AND INTELLIGENT SYSTEMS 2:(2) pp. 95-100. (2010)

S2. György Pilászy, György Rácz, Péter Arató,"Communication Time Estimation in High Level Synthesis", Periodica Polytechnica, 2012, lektorálás és megjelenés folyamatban

S3. György Pilászy**,** György Rácz, Péter Arató, **„**The effect of increasing the latency time in High Level Synthesis", Periodica Polytechnica, lektorálás és megjelenés folyamatban

Further journal papers and book chapters

S4. Pilászy György, Móczár Géza, "Angol-magyar informatikai szótár", a számítógép architektúrák és digitális rendszerek témakör könyv fejezeteinek elkészítése, Budapest: Tinta Könyvkiadó, 2006. pp. 1-384., ISBN: 963 9372 79 X

S5. Pilászy György, Horváth Tamás, "Digitális technika feladatok és megoldásuk", Elektronikus formában közzétett egyetemi jegyzet az MSC villmosmérnök szakra jelentkező hallgatók számára

S6. Pilászy György, "Gyakorlati anyagok a Beágyazott irányító rendszerek tantárgy gyakorlataihoz.: A CAN és LIN protokollok", Elektronikusan közzétett egyetemi jegyzet

S7. Pilászy György, Horváth Tamás, "Digitális technikai alapmérések", Jegyzetszám: VI205010, 2011, BME Viking Zrt

Conferences

S8. Pilászy György, Móczár Géza, "Moduláris mikrokontrolleres rendszerek távoli felügyelete", In: Measurement and Automation. Miskolc, Magyarország, 2001.03.01-2001.03.02. pp. 45-50. Paper, Section F:Measurement and Automation, microCAD 2001, Miskolc, 2001

S9. Pilászy György, Móczár Géza, "Modular Microcontroller System", In: Lehoczky László, Kalmár László (szerk.), MicroCAD 2004 International Scientific Conference. Miskolc, Magyarország, 2004.03.18-2004.03.19. Miskolc: Miskolci Egyetem Innovációs és Technológia Transzfer Centruma, pp. 81-86. Vol. G, Automation and telecommunication (Automatizálás és telekommunikáció) (ISBN: 963 661 615 9)

S10. Pilászy György, Komáromi Zoltán, Móczár Géza, "Központi irányítóegység moduláris, elosztott intelligenciájú irányítástechnikai rendszerekhez", In: Bíró Károly Ágoston Sebestyén-Pál György (szerk.), IX. ENELKO - XVIII. SámOkt Nemzetközi energetikai-elektrotechnikai és számítástechnikai konferencia. Csíksomlyó, Románia, 2008.10.09-2008.10.12. (EMT ; Babes-Bolyai University) Kolozsvár: Erdélyi Magyar Műszaki Tudományos Társaság, pp. 172-178. Paper &. Vol. 1

S11. Pilászy György, Komáromi Zoltán, Móczár Géza, "Elosztott irányítórendszer napkollektoros enegetikai rendszerekhez", In: Bíró Károly Ágoston, Sebestyén-Pál György (szerk.), IX. ENELKO - XVIII.SzámOkt Nemzetközi energetikai-elektrotechnikai és számítástechnikai konferencia. Csíksomlyó, Románia, 2008.10.09-2008.10.12. (EMT ; Babes-Bolyai University) Kolozsvár: Erdélyi Magyar Műszaki Tudományos Társaság, pp. 220-225. Paper &. Vol. 1

S12. Pilászy György, Komáromi Zoltán, Móczár Géza, "A method for designing and installing distributed intelligent, embedded systems", In: ISW1 International Scientific Workshop on DCS 1th Meeting: &. Lillafüred, Magyarország, 2010.10.25-2010.10.27. Miskolc-Lillafüred: University of Miskolc, pp. 58-65. Paper, (Proceedings of 1st International Scientific Workshop on DCS; 1.) Vol. 1, Proceedings of 1st International Scientific Workshop on DCS (ISBN: 978-963-661-950-3) This publication is supported by the Hungarian Section of IEEE

S13. Pilászy György, Móczár Géza, Rácz György, "Microcontroller based variable-speed low-power induction motor drive", In: Dr Bíró Károly Ágoston, Dr Sebestyén-Pál György (szerk.)ENELKO2011 XII. Nemzetközi Energetika-Elektrotechnika konferencia. Kolozsvár, Románia, 2011.10.06-2011.10.09. Kolozsvár: pp. 79-90. Paper 13. (ISBN: 1842-4546)

S14. Rácz György, Pilászy György, Móczár Géza, "Efficient microcontroller based methods for single phase induction motor control", In: Ildikó BÖLKÉNY (szerk.), Proceedings of 2nd International Scientific Workshop on DCS. Miskolc, Magyarország, 2011.10.24-2011.10.26. Miskolc: pp. 26-33. Paper 3. (ISBN: &)

Other publications

S15. Pilászy György, "Programozható hő- és villamos energia kapcsoló berendezés", (1998) TDK

S16. Pilászy György, Móczár Gergő, "Beágyazott rendszerek távoli felügyelete", (2000) TDK

S17. Pilászy György, Móczár Gergő, "Távoli eléréssel rendelkező intelligens mikrokontrolleres vezérlő", (2000) TDK

S18. Pilászy György, Király Pál, "Villamos kisülések gázokban", (2000) TDK

S19. Pilászy György, "Távoli elérésű intelligens mikrokontrolleres vezérlő", (2002), Diplomamunka

Research report (internal)

S20. Móczár Géza, Pilászy György, Nagy Ákos, "Intelligens, elosztott, öntanuló irányítási rendszer kialakítása napenergiás és vegyes rendszerekhez.: GVOP-3.1.1.-2004.05.0515/3.", pp. 1-170., (2005)

S21. Arató Péter, Móczár Géza, Pilászy György és társai, "Feladatfüggő felépítésű többprocesszoros célrendszerek szintézis algoritmusainak kutatása.: OTKA T72611.", (2011)

References

[1] CAN specification 2.0A (http://www.can-cia.org/index.php?id=441)

[2] CAN specification 2.0B (http://www.can-cia.org/index.php?id=441)

[3] Péter Arató, Tamás Visegrády, István Jankovits: High Level Synthesis of Pipelined Datapaths, John Wiley & Sons, New York, ISBN: 0 471495582 4, 2001

[4] I2C-bus specification and user manual Rev. 4 — 13 February 2012 (http://www.nxp.com/documents/user_manual/UM10204.pdf)

[5] Pilászy György, Móczár Géza, Remote control of modular microcontroller systems, Microcad 2001, In: Measurement and Automation. Miskolc, Hungary, 2001.03.01-2001.03.02. pp. 45-50.

[6] Michel Goraczko, Jie Liu, Dimitrios Lymberopoulos: Energy-Optimal Software Partitioning in Heterogeneous Multiprocessor Embedded Systems, DAC 2008, June 8–13, 2008, Anaheim, California, USA

[7] P. Arató, D. Drexler, G. Kocza, G. Suba: Synthesis of a Task-dependent Pipelined Multiprocessing Structure, submitted to the ACM Transactions on Design Automation of Electronic Systems

[8] The I2C Bus specification version 2.1, January 2000, http://www.nxp.com/documents/other/39340011.pdf

[9] Philippe Coussy, Daniel D. Gajski, Michael Meredith, Andres Takach, An Introduction to High-Level Synthesis, IEEE Design & Test of Computers, 2009

[10] 8251A Programmable communication interface, 1993, Intel Corporation, Document number: 205222-003

[11] IEC/EN60730 Safety Standard

[12] Arató Péter, Logikai rendszerek tervezése, BME Printer kft, 2011

[13] H. Peterson, Introduction to switching theory and logical design, Wiley, 1968

[14] D. Lewin, Logical design of switching circuits, Nelson, 1968

[15] Raul Camposano, Wayne Wolf, High-Level VLSI Synthesis, Kluwer, 1991

[16] Claude Baron, Jean-Claude Geffroy, Gilles Motet, Embedded System Applications, Kulwer, 1997

[17] Jiang Xu, Wayne Wolf, A Design Methodology for Application-Specific Networks-on-Chip, ACM Transactions on Embedded Computing Systems, Vol. 5, No. 2, May 2006, Pp 263–280

[18] Kandár Tibor Hierarchikus rendszer szintű szintézis, doktori értekezés, 2007, BME

[19] CORPORATE Inc. Institute of Electrical and Electronics Engineers. IEEE Standard Description, Language Based on the VERILOG Hardware Description Language, 1364-1995. IEEE Standards, Office, New York, NY, USA, 1996.

[20] IEEE. IEEE Standard VHDL Reference Manual. IEEE, 1988.

[21] T. Kandár. Systematic VHDL code generation based on high level synthesis results. In INES2003, The 7th IEEE International Conference On Intelligent Engineering Systems, pages 716–720, Assiut, Luxor, Egypt, March 3–7 2003. ISBN:977.246.048.3, ISSN:1562-5850.

[21] Ahmed A. Jerraya, Jean Mermet, System-Level Synthesis, NATO science Series E, Applied Sciences – Vol. 357, Kluwer, 1999

[22] Zoltán Ádám Mann, András Orbán, Péter Arató (dec 2007) Finding optimal hardware/software partitions. Form. Methods Syst. Des. 31 (3) pp. 241–263. Kluwer Academic Publishers. Hingham, MA, USA.

[23] P. Arato, S. Juhasz, Z.A. Mann, A. Orban, D. Papp (sept. 2003) Hardware-software partitioning in embedded system design. In Intelligent Signal Processing, 2003 IEEE International Symposium on. pp. 197 - 202.

[24] Péter Arató, Zoltán Ádám Mann, András Orbán (jan 2005) Algorithmic aspects of hardware/software partitioning. ACM Trans. Des. Autom. Electron. Syst. 10 (1) pp. 136–156. ACM. New York, NY, USA.

[25] High-Level Synthesis Blue Book, 2010, Mentor Graphics Corporation

[26] Gergely Suba, Hierarchical pipelining of nested loops in high-level synthesis, 2013, Periodica Polytechnica, lektorálás és megjelenés folyamatban

[27] Gál Tibor: Interfésztechnikák, Szak kiadó, 2010, ISBN 978-963-9863-13-2

[28] USB 2.0 Specification, (http://www.usb.org/developers/docs)

[29] Andrew S. Tannembaum: Számítógép hálózatok, Panem, 1999

[30] BME-PS CAN plus protokoll specifikáció, BME-IIT, 2006

[31] Junwei Hou, Wayne Wolf: Process Partitioning for Distributed Embedded Systems, proceedings of the 4th International workshop on hardware/software co-design (Codes/CASHE '96), 0-8186-7243-9/96, 1996, IEEE

[32] Jiang Xu, Wayne Wolf, A Design Methodology for Application-Specific Networks-on-Chip, ACM Transactions on Embedded Computing Systems, Vol. 5, No. 2, May 2006, Pages 263-280