結 論

本論文では，高速動作するディジタル回路の論理的設計手法に関する研究と，通信・画像処理・演算回路への応用について述べ，ディジタル回路における，高速性，信頼性，動作の正しさ等の検証に関する，数々の方策を提案した．本論文で述べたことを以下にまとめる．

第1章では，セルオートマトンの概念を用いた信頼性の高い通信用バッファ回路の提案とゲートアレイへの実装について述べた．このバッファは一過性の故障における耐故障性を有しており，通信ネットワークにおけるハードウェアレベルでの信頼性向上に寄与する．

第2章では，パイプライン型並列システムの仕様をペトリネットでモデル化し，設計時にシステムの安定性を検討することについて述べた．実システムをモデル化する際，従来のペトリネットを使用するとネット規模が増大し困難である．ここでは，ネット要素を拡張し，記述能力を改善した論理カラーペトリネット（LC-Petri net）を提案した．LC-Petri net で仕様を記述することにより，実システムの安定性・デッドロック状態の有無の検出が可能となり，並列システムの信頼性確保に寄与する．

第3章では，高速画像処理システムの構築と新たに提案する2値画像向け符号化方式について述べた．この画像処理装置はプリント基板等の表面検査を行うものである．検査のマッチング処理を高速かつリアルタイムに行うため，これらの性質を兼ね揃えた復号方式を考える必要があった．このため，2進桁圧縮符号化方式（Binary Place Coding method: BPC）を提案し，この符号を高速リアルタイム復号可能なパイプライン型復号ハードウェアを設計，ゲートアレイへ実装した．

第4章では，演算回路の数学定義と動作の証明について述べた．ハードウェア回路の設計誤り（バグ）が及ぼすシステムへの影響は計り知れない．従来のハードウェア設計検証法の多くは，論理ゲート間の遅延時間がゼロまたは適当な値を仮定した上でのシミュレーション手法に留まっている．本論文では，論理演算子，演算子とハードウェアゲートとの関係，ゲート同士の信号線による接続等の数学定義をもとに，2の補数演算回路を例とし，演算回路が正しく動作することを数学的に証明した．

以上の各章で提案した，ディジタル回路における，高速性，信頼性，動作の正しさ等を確保する，論理的な回路設計手法は，今後のディジタル回路設計における基礎的技術として，重要な資料となると考えている．

半導体技術のさらなる発展に伴い，ディジタル回路は今後ますます複雑化かつ高速動作するようになり，種々の産業へ様々な形で応用・浸透していくであろう．このような状況の中で，今後，本研究の成果が産業分野で貢献することができれば，筆者にとって幸せである．

謝 辞

本研究を開始する機会を与えて下さり, 本論文をまとめるにあたり, 信州大学工学部情報工学科 中村八束 教授 には, 終始熱心な御指導, 御教示を賜わりました. ここに深甚の謝意を表し, 厚く御礼申し上げます.

本研究に関し, 常に懇切なる御指導ならびに御助言を賜わった, 信州大学工学部情報工学科 不破 泰 助教授 に, 厚く御礼申し上げます.

本論文をまとめるにあたり, 数々の御教示と御助言を賜わった, 信州大学工学部情報工学科 師玉康成 教授, 同学部数学教室数理工学講座 酒井雄二 教授 に厚く御礼申し上げます.

本研究に関し, 数々の御教示と御助言を賜わった, 東京商船大学商船学部交通電子制御工学科 江口正義 助教授 に, 厚く御礼申し上げます.

本研究に関し, 熱心な御討論を頂いた, 信州大学工学部情報工学科 Pauline N. Kawamoto (Aiura) 助手 に, 厚く感謝致します.

本研究に関し, 数々の御教示を頂きました, Warsaw 大学 Andrzej Trybulec 教授 , ならびに Mizar プロジェクトメンバーの方々に, 厚く感謝致します.

本研究にあたり, 信州大学工学部情報工学科 堀内美恵子 技官, ならびに, 同情報基礎講座の方々には, 多大なる御助力を頂きました. ここに深く感謝いたします.

長野工業高等専門学校 森 粛 校長, 電子制御工学科 坂口正雄 教授, 芳賀 武 教授, 岸佐年 教授, 森山 実 助教授, 服部 忍 助教授, 鈴木 宏 講師, 中島隆行 講師, 小野伸幸 講師, 堀内富雄 助手, 加藤正幸 技官, ならびに, 同高等専門学校の方々には, 日頃より多大なる御支援を賜わりました. 心より御礼申し上げます.

参考文献

[1] 不破，中村，清水：“2重のリング構造から成るキャンパスネットワーク – S-net について –”，電子情報通信学会論文誌 (B)，Vol.J71-B, No.12，pp.1672-1681 (1988).

[2] 不破，中村：“ウォッチドッグタイマの有効性に関する統計的考察”，電子情報通信学会論文誌 (D)，Vol.J71-D, No.11，pp.2414-2423 (1988).

[3] 不破，中村：“リング型ネットワークにおけるノードのハングアップ検出と復旧法について”，電子情報通信学会論文誌 (B-I)，Vol.J72-B-I, No.9, pp.730-740 (1989).

[4] “High Performance CMOS DATA BOOK SUPPLEMENT 1990”，*Integrated Device Technology* (1990).

[5] 川中，角山，内藤：“二次元セルラー・オートマトンに基づくフォールト・トレラント並行処理システム”，電子情報通信学会秋季大会講演論文集, Vol.6，pp.327-328 (1991).

[6] “Fast CMOS Data Book July 1990”，*LOGIC DEVICES INCORPORATED* (1990).

[7] 大場, 村田, 宮原：“ ATM 網におけるバースト性を持つトラヒックのフロー抑制制御”，電子情報通信学会技術研究報告，SSE90-131，IN90-92，(1991).

[8] “MSM6252RS FIFO Memory DATA SHEET”，沖電気工業（株） (1986).

[9] “PAL Device Data Book Bipolar and CMOS”，*Advanced Micro Devices* (1990).

[10] “PALASM User's Manual”，*Advanced Micro Devices* (1991).

[11] Saito : " Optimal Control of Variable Rate Coding in Integrated Voice/Data Packet Networks" , 電子情報通信学会技術研究報告 , IN87-108 , (1988).

[12] 田邉, 畝本, 松尾 : " レジスタ挿入型優先制御のスロットリング LAN への応用" , 電子情報通信学会技術研究報告 , IN89-11 , (1989).

[13] "TPC10 and TPC12 Series FPGA ALES1, Release 2.1 User's Guide for PC" , *Texas Instruments Incorporated* (1992).

[14] "TPC10 and TPC12 Series FPGA Action Logic System (TI-ALS), Release 2.1 User's Guide" , *Texas Instruments Incorporated* (1991).

[15] "TPC10 Series Field-Programmable Gate Arrays Data Sheet" , *Texas Instruments Incorporated* (1991).

[16] 戸出, 山本, 岡田, 手塚 : " 放送型伝送路を用いた制御先行型 ATM 交換機構成法" , 電子情報通信学会技術研究報告 , SSE90-39 , IN90-32 , CS90-28 (1990).

[17] Tsunoyama, Naito : "Fault-tolerant parallel processor based on a linear cellular automaton model" , *Systems and Computers in Japan*, Vol.21 No.14 , pp.23-31 (1990).

[18] "WORKview Series I Vol1,2 : Version A January 1990, For use with Workview 4.0" , *Viewlogic Systems Inc.* (1990).

[19] 和崎, 不破, 江口, 中村: "セルオートマトンの概念を用いた自己回復能力をもつ通信用バッファ", 電子情報通信学会技術研究報告, CAS92-102, NLP92-82, pp.61-68 (1993).

[20] 和崎, 脇本, 不破, 江口, 中村 : "FPGA を用いた自己回復能力をもつ通信用バッファ", *The 1st Japanese FPGA/PLD Design Conf.*, Vol.1, No.28, pp.98-111 (1993).

[21] 和崎, 不破, 江口, 中村 : "セルオートマトンの概念を用いた自己回復能力をもつ通信用バッファ", 電子情報通信学会論文誌 (D-I), Vol.J77-D-I, No.1, pp.41-52 (1994).

[22] K.Wasaki, Y.Fuwa, M.Eguchi, Y.Nakamura : "A Self-Recovering Communication Buffer Based on the Concept of Cellular Automaton", *Systems and Computers in Japan*, Vol.25, No.8, pp.1-15 (1994).

[23] K.Jensen : "Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use", Vol.1, Basic Concepts, *Springer-Verlag* (1992).

[24] P.N.Kawamoto, M.Eguchi, Y.Fuwa, Y.Nakamura : "Deadlocks/Traps which result from the Order-Dependence of Petri Net Transitions", *Technical Report of IEICE*, SS92-29, KBSE92-50, pp.345-346 (1993).

[25] P.N.Kawamoto, M.Eguchi, Y.Fuwa, Y.Nakamura : "Introduction of the Concept of Order in Petri Net Evaluation and on the Verification of Systems", *Technical Report of IEICE*, CAS93-70, pp.109-166 (1993).

[26] P.N.Kawamoto, Y.Fuwa, Y.Nakamura : "Basic Concepts for Petri Nets with Boolean Markings" , *Journal of Formalized Mathmatics*, Vol.4, No.1 (1993).

[27] P.N.Kawamoto: "Applications of Petri Net Theory in the Design of Hardware/Software Systems", *Ph.D. dissertation, Shinshu University, Nagano* (1996).

[28] M.Kitazawa, P.N.Kawamoto, Y.Fuwa, Y.Nakamura : "Petri Net Software Development for Parallel/Distributed Systems", *Proc. Parallel and Distributed Computing and Systems(PDCS'96)*, pp.418-420 (1996).

[29] 李, 不破, 江口, 中村 : "ペトリネットを用いたハードウェア設計とその高速化について" , 電子情報通信学会技術研究報告 , CAS92-103 , NLP92-83, pp.69-76 (1993).

[30] 村田 : "ペトリネットによるシステムの設計と解析", 共立出版 (1981).

[31] T.Murata : "Petri Nets: Properties, Analysis and Applications ", *Proc. IEEE*, Vol.77, No.4, pp.541-580 (1989).

[32] 村田, 辻 : "ペトリネットによる並行処理プログラムの解析手法", 情処誌, Vol.34, No.6, pp.701-709 (1993).

[33] Y.Nagao, H.Ohta, H.Urabe, S.Kumagai : "Petri Net Based Programming System for FMS",*IEICE Trans. Fundamentals.*, Vol.E75–A, No.10, pp.1326–1334 (1992).

[34] T.Nakao, T.Yamagishi, P.N.Kawamoto, Y.Fuwa, Y.Nakamura: "A Petri Net Based Protocol Design Tool for Wireless Data Communication", *Proc. Multi-Dimensional Mobile Communications 96*, pp.307-311 (1996).

[35] J.L.Peterson : "Petri Net Theory and the Modeling of Systems"(Japanese edition), *Kyouritsu Publishing Co., Inc.* (1981).

[36] W.Reisig : "Petri Nets"(Japanese edition), *Springer–Verlag Tokyo* (1985).

[37] 和崎, 不破, 江口, 中村 : "制御ソフトウェアに適したペトリネットの拡張" , 電子情報通信学会技術研究報告 , COMP93–12 , SS93–6, pp.37–44 (1993).

[38] 和崎, 不破, 江口, 中村 : "CASE ツールとしての論理カラーペトリネット (LC–net) の能力", 電子情報通信学会技術研究報告 , CAS93–69, pp.101–108 (1993).

[39] K.Wasaki, Y.Fuwa, M.Eguchi, Y.Nakamura : "Logical Coloured Petri Net Expanded to be Suitable making the Control System Model", *Fourth International Conference on Control, Automation, Robotics and Vision (ICARCV'96) Proc.*, TA-2-2, pp.708–713 (1996).

[40] 安部井, 黒須, 他: "画像圧縮伸長の高速化方式", 情報処理学会第 48 回全国大会講演論文集, No.2, pp.359 (1994).

[41] "The FPGA Data Book and Design Guide : ACT-1 family" , *Actel Corporation* (1996).

[42] 原島: "画像情報圧縮", オーム社 (1991).

[43] 長谷, 宮沢, 他: "ウィンドウ調整回路内蔵高速符号・復号 LSI の開発", 電子情報通信学会秋季大会講演論文集, C-551, pp.219 (1990).

[44] T.S.Huang: "Coding of Two-Tone Images" , *IEEE Trans. Commun.*,COM-25,11,pp.1406-1424 (1977).

[45] D.A.Huffman : "A Method for the Construction of Minimum Redundancy Codes " , *Proc. IRE*, 40, 10, pp.1098-1101 (1952).

[46] R.Hunter et al.: "International Digital Facsimile Coding Standards", *Proc. IEEE*, 68, 7, pp.854 - 867 (1980).

[47] 井原 : "リアルタイムシステムとは", 情報処理学会誌, 35, 1, pp.12-17 (1994).

[48] ITU-R Rec. T.81 : "Digital Compression and Coding of Continuous-tone Still Image" (JPEG), (1992).

[49] ITU-R Rec. T.82 : "Progressive Bi-level Image Compression" (JBIG), (1992).

[50] 勝野, 小池, 他: "JBIG コーデックと画像通信システムの開発", 画像電子学会第 21 回年次大会・Visual Computing '93 予稿集, pp.21-24 (1993).

[51] S.Kim, J.Lee et al.: "A New Chain-Coding Algorithm for Binary Images Using Run-Length Codes", *Comp. Vision Graphics Image Proc.*, 41,1,pp.114-128 (1988).

[52] 宮川: "情報理論", 電子通信大学講座 (1979).

[53] 中山, 末広, 他: " Run-Length 符号化による帯域圧縮の一方式" , テレビジョン学会第 15 回録画磁気研究会資料 15-1 (1970).

[54] K.Nakamura, Y.Kojima et al: "High Speed Encoding and Decoding Processor for Group 4 Facsimile Aparatus" *IEEE ICC'84*, pp.219-222 (1984).

[55] 佐藤, 村山, 他: " 高速 2 値イメージ圧縮伸長 LSI の開発" , 電子情報通信学会技術研究報告, Vol.88, No.255, pp.51-58 (1988).

[56] 山崎, 小野, 他: "2 値画像階層型符号化方式-JBIG アルゴリズム", 画像電子学会誌,20,1,pp.41-49(1991).

[57] 柳谷, 上野, 他: "適応予測を用いた多値画像のロスレス符号化", 画像電子学会研究会予稿, 95-04-01,pp.1-4 (1995).

[58] 安田: "マルチメディア符号化の国際標準", 丸善 (1991).

[59] 和崎, 不破, 江口, 中村："逐次リアルタイム復号処理が可能な 2 値画像圧縮法", 電子情報通信学会技術研究報告, IT95-4, pp.18-24 (1995).

[60] 和崎, 不破, 江口, 中村："画像処理に適した高速リアルタイム復号が可能な 2 値画像符号とその評価", 画像電子学会誌, Vol.25, No.6, pp.734–742 (1997).

[61] 市川, 加藤, 後藤："循環パイプラインアーキテクチャー Pipelined MIMD の試み", 情報処理学会第 37 回全国大会講演論文集, No.4, N-4 (1988).

[62] 石浦, 高橋, 矢島："論理回路の正確なタイミング検証のための時間記号シミュレーション", 情報処理学会論文誌, Vol.31, No.12, pp.1832-1839 (1990).

[63] L.Kleeman, A.Cantoni : "Can Redundancy and Masking Improve the Performance of Synchronizers?", *IEEE Trans. Comput.*, Vol.C-35, No.7, pp.643-646 (1986).

[64] 中村, 西山, 不破 : "多段順序回路における信号伝搬のタイミング検証システム", 電子情報通信学会論文誌 (A) , Vol.J75-A, No.12, pp.1826-1836 (1992).

[65] 西山, 不破, 江口, 中村 : "クロックモデルによるデジタル回路のタイミング検証法", 電子情報通信学会技術研究報告 , CAS93-65, pp.73-80 (1993).

[66] 小倉："CPLD/FPGA の開発手法", The First Japanese FPGA/PLD Design Conference 技術講座 4, pp.46-62 (1993).

[67] K.E.Stoffers : "Test Sets for Combinational logic – The Edge – Tracing Approach ", *IEEE Trans. Comput.*, Vol.C-29, No.8, pp.741-746 (1980).

[68] C.E.Strangio : "DIGITAL ELECTRONICS: Fundamental Concepts and Applications", *Prentice-Hall Inc.*, pp.355-385 (1980).

[69] S.H.Unger, C.J.Tan : "Clocking Schemes for High-Speed Digital Systems", *IEEE Trans. Comput.*, Vol.C-35, No.10, pp.880-895 (1986).

[70] E.Bonarska: "An Introduction to PC Mizar", *Mizar Users Group, Fondation Philippe le Hodey*, Brussels (1990).

[71] A.Trybulec and P.Rudnicki: "A Collection of TEXed Mizar Abstracts", *University of Alberta*, Canada (1989).

[72] A.Trybulec : "Tarski Grothendieck set theory", *Journal of Formalized Mathematics*, 1 (1989).

[73] G.Bancerek : "The fundamental properties of natural numbers", *Journal of Formalized Mathematics*, 1 (1989).

[74] G.Bancerek : "The ordinal numbers", *Journal of Formalized Mathematics*, 1 (1989).

[75] G.Bancerek and K.Hryniewiecki : "Segments of natural numbers and finite sequences", *Journal of Formalized Mathematics*, 1 (1989).

[76] A.Trybulec : "Binary operations applied to functions", *Journal of Formalized Mathematics*, 1 (1989).

[77] A.Trybulec : "Enumerated sets", *Journal of Formalized Mathematics*, 1 (1989).

[78] C.Bylinski : "Binary operations", *Journal of Formalized Mathematics*, 1 (1989).

[79] C.Bylinski : "Functions and their basic properties", *Journal of Formalized Mathematics*, 1 (1989).

[80] C.Bylinski : "Functions from a set to a set", *Journal of Formalized Mathematics*, 1 (1989).

[81] C.Bylinski : "Partial functions", *Journal of Formalized Mathematics*, 1 (1989).

[82] A.Darmochwai : "Finite sets", *Journal of Formalized Mathematics*, 1 (1989).

[83] A.Trybulec : "Tuples, projections and Cartesian products", *Journal of Formalized Mathematics*, 1 (1989).

[84] K.Hryniewiecki : "Basic properties of real numbers", *Journal of Formalized Mathematics*, 1 (1989).

[85] E.Woronowicz : "Relations and their basic properties", *Journal of Formalized Mathematics*, 1 (1989).

[86] E.Kusak, W.Leonczuk, and M.Muzalewski : "Abelian groups, fields and vector spaces", *Journal of Formalized Mathematics*, 1 (1989).

[87] Z.Trybulec and H.Swieczkowska : "Boolean properties of sets", *Journal of Formalized Mathematics*, 1 (1989).

[88] M.J.Trybulec : "Integers", *Journal of Formalized Mathematics*, 2 (1990).

[89] W.A.Trybulec : "Pigeon hole principle", *Journal of Formalized Mathematics*, 2 (1990).

[90] E.Woronowicz : "Many-argument relations", *Journal of Formalized Mathematics*, 2 (1990).

[91] C.Bylinski : "The modification of a function by a function and the iteration of the composition of a function", *Journal of Formalized Mathematics*, 2 (1990).

[92] C.Bylinski : "Finite sequences and tuples of elements of a non-empty sets", *Journal of Formalized Mathematics*, 2 (1990).

[93] G.Bancerek : "Konig's theorem", *Journal of Formalized Mathematics*", 2 (1990).

[94] G.Bancerek : "Tarski's classes and ranks", *Journal of Formalized Mathematics*, 2 (1990).

[95] A.Lango and G.Bancerek : "Product of families of groups and vector spaces", *Journal of Formalized Mathematics*, 4 (1992).

[96] T.Nishiyama and Y.Mizuhara : "Binary arithmetics", *Journal of Formalized Mathematics*, 5 (1993).

[97] A.Trybulec : "Many-sorted sets", *Journal of Formalized Mathematics*, 5 (1993).

[98] A.Trybulec : "Many sorted algebras", *Journal of Formalized Mathematics*, 6 (1994).

[99] Y.Nakamura, P.Rudnicki, A.Trybulec and P.N.Kawamoto : "Preliminaries to circuits, I.", *Journal of Formalized Mathematics*, 6 (1994).

[100] Y.Nakamura, P.Rudnicki, A.Trybulec and P.N.Kawamoto : "Preliminaries to circuits, II.", *Journal of Formalized Mathematics*, 6 (1994).

[101] Y.Nakamura and G.Bancerek : "Combining of circuits", *Journal of Formalized Mathematics*, 6 (1994).

[102] Y.Nakamura, P.Rudnicki, A.Trybulec and P.N.Kawamoto : "Introduction to circuits, I.", *Journal of Formalized Mathematics*, 6 (1994).

[103] Y.Nakamura, P.Rudnicki, A.Trybulec and P.N.Kawamoto : "Introduction to circuits, II.", *Journal of Formalized Mathematics*, 7 (1995).

[104] G.Bancerek and Y.Nakamura : "Full adder cucuit. Part I.", *Journal of Formalized Mathematics*, 7 (1995).

[105] K.Wasaki and P.N.Kawamoto : "2's Complement Circuits. Part I.", *Journal of Formalized Mathematics*, 8 (1996).

研究業績

論文

[1] 和崎 克己, 脇本 直樹, 不破 泰, 江口 正義, 中村 八束："FPGA を用いた自己回復能力をもつ通信用バッファ", *The 1st Japanese FPGA/PLD Design Conference*, Vol.1, No.28, pp.98-111 (Jul.,1993).

[2] 和崎 克己, 不破 泰, 江口 正義, 中村 八束："セルオートマトンの概念を用いた自己回復能力をもつ通信用バッファ", 電子情報通信学会論文誌 D-I, Vol.J77-D-I, No.1, pp.41-52 (Jan.,1994).

(Katsumi Wasaki, Yasushi Fuwa, Masayoshi Eguchi and Yatsuka Nakamura: "A Self-Recovering Communication Buffer Based on the Concept of Cellular Automaton", *Systems and Computers in Japan*, Vol.25, No.8, pp.1-15 (Oct.,1994))

[3] Katsumi Wasaki, Yasushi Fuwa, Masayoshi Eguchi and Yatsuka Nakamura: "Logical Coloured Petri Net Expanded to be Suitable making the Control System Model", *Fourth International Conference on Control, Automation, Robotics and Vision (ICARCV'96) Proc.*, TA-2-2, pp.708-713 (Dec.,1996).

[4] 和崎 克己, 不破 泰, 江口 正義, 中村 八束："画像処理に適した高速リアルタイム復号が可能な 2 値画像符号とその評価", 画像電子学会誌, Vol.25, No.6, pp.734-742 (Jan.,1997).

[5] Katsumi Wasaki and Pauline N. Kawamoto: "2's Complement Circuit.", *Journal of FORMALIZED MATHEMATICS*, Vol.8 (Jan.,1997).

報 文

[1] 清水 英孝，和崎 克己："計測画像の転送のためのデータ圧縮解凍技術（第 2 報）～動画像データのトラフィック解析～", 長野県情報技術試験場研究報告，No.11, pp.20-23 (Aug.,1995).

[2] 和崎 克己，不破 泰，江口 正義，中村 八束："制御システムのモデル化に適するように拡張した論理カラーペトリネット"，長野工業高等専門学校紀要, 第 29 号, pp.17-25 (Dec.,1995).

[3] 清水 英孝，和崎 克己："計測画像の転送のためのデータ圧縮解凍技術（第 3 報）～スロットリングにおけるマルチメディアモデルの検討～", 長野県情報技術試験場研究報告，No.12, pp.82-83 (Aug.,1996).

[4] 和崎克己，不破 泰，中村八束, 清水英孝："パケット処理装置に適したトークン方式バス調停法と性能評価"，長野工業高等専門学校紀要, 第 30 号, pp.63-74 (Dec.,1996).

[5] 小野伸幸, 和崎克己, 鈴木 宏, 堀内富雄, 岸 佐年, 坂口正雄："高専電子制御工学科における実験実習指導"，工学教育, Vol.45, No.1, pp.21-25 (Jan.,1997).

学会発表

[1] 和崎克己，不破 泰，江口正義，中村八束："拡張ペトリネットを用いた制御ソフトウェア開発"，電子情報通信学会信越支部大会講演論文集，No.172, pp.343-345 (Oct.,1992).

[2] 和崎克己，不破 泰，江口正義，中村八束："セルオートマトンの概念を用いた自己回復能力をもつ通信用バッファ"，電子情報通信学会技術研究報告（回路とシステム研究会），CAS92-102，NLP92-82, pp.61-68 (Jan.,1993).

[3] 和崎克己，不破 泰，江口正義，中村八束："制御ソフトウェアに適したペトリネットの拡張"，電子情報通信学会技術研究報告（ソフトウェアサイエンス研究会），COMP93-12，SS93-6, pp.37-44 (May ,1993).

[4] 和崎 克己, 脇本 直樹, 不破 泰，江口 正義，中村 八束："FPGA を用いた自己回復能力をもつ通信用バッファ"，*The 1st Japanese FPGA/PLD Design Conference*, Vol.1，No.28, pp.98-111 (Jul.,1993).

[5] 和崎克己，不破 泰，江口正義，中村八束："CASE ツールとしての論理カラーペトリネット（LC-net）の能力"，電子情報通信学会技術研究報告（回路とシステム研究会），CAS93-69, pp.101-108 (Sep.,1993).

[6] 古澤敦夫，和崎克己，中村八束："バックラッシやたわみを考慮したマニュピレータ制御の方式について"，電子情報通信学会信越支部大会講演論文集，No.1, pp.1-2 (Oct.,1994).

[7] 斎藤 剛，和崎克己，不破 泰："2 重トークンリング方式を用いたパケット交換機の性能解析"，電子情報通信学会信越支部大会講演論文集，No.60, pp.119-120 (Oct.,1994).

[8] 大澤竜馬，和崎克己，中村八束："コントローラシステム設計の為のペトリネットを用いた手法"，電子情報通信学会信越支部大会講演論文集，No.179 pp.357-358 (Oct.,1994).

[9] 和崎克己，不破 泰，江口正義，中村八束："逐次リアルタイム復号処理が可能な2値画像圧縮法"，電子情報通信学会技術研究報告（情報理論研究会），IT95-4, pp.19-24 (May ,1995).

[10] 和崎克己，不破 泰，中村八束，斎藤 剛，松下忠嗣："パケット処理装置に適したトークン方式バス調停法について"，電子情報通信学会信越支部大会講演論文集, R8, pp.469-470 (Oct.,1995).

[11] 小野伸幸, 和崎克己, 岸 佐年, 鈴木 宏, 堀内富雄, 坂口正雄："高専電子制御工学科における実験実習指導～システム技術者能力の育成を目指す総合実験実習の導入～"，日本工学教育協会工学・工業教育研究講演会講演論文集, 教育研究指導-IV, No.86 , pp.271-274 (Jul.,1996).

[12] 和崎克己, 不破 泰, 中村八束, 斎藤 剛, 松下忠嗣："パケット交換型リングネットワークのノード構成に適したバス調停法"，日本産業技術教育学会第39回全国大会講演論文集, No.207 pp.36 (Jul.,1996).

[13] 和崎克己, 不破 泰, 江口正義, 中村八束："制御システムのモデル化に適するように拡張した高レベルペトリネット"，日本産業技術教育学会第39回全国大会講演論文集, No.208 pp.37 (Jul.,1996).

[14] 和崎克己, 不破 泰, 江口正義, 中村八束："制御システムのモデル化に適するように拡張した論理色付きペトリネット"，電気学会 電子・情報・システム部門大会講演論文集, B-7-1, pp.411-416 (Sep.,1996).

[15] 平出桂介, 植木竜暁, 和崎克己："複数のロボットで構成されたFAシステムのペトリネットモデルと実装"，電子情報通信学会信越支部大会講演論文集, D4, pp.175-176 (Sep.,1996).

[16] Katsumi Wasaki, Yasushi Fuwa, Masayoshi Eguchi and Yatsuka Nakamura: "Logical Coloured Petri Net Expanded to be Suitable making the Control System Model", *Fourth International Conference on Control, Automation, Robotics and Vision (ICARCV'96) Proc.*, TA-2-2, pp.708-713 (Dec.,1996).

付録 A　FPGA を用いた FIFO メモリの実装

以下, FPGA(Field-Programmable Gate Arrays) [15] を用いたFIFO メモリのハードウェアへの実装，FPGA を採用した理由，実装例，及びその使用感について述べる [20].

A.1　FPGA を採用した理由

FIFO メモリをハードウェア化するにあたり，FPGA を採用した．FPGA を採用した理由を以下に挙げる．

(1) 小規模 PLD に比較して容量・入出力ピン数ともに多い

従来，小規模 PLD(=Programmable Logic Device) [9] を用いたハードウェアの実装を行っていたが，その容量や入出力ピン数の制限が多かった．本メモリの待ち行列管理用セルを小規模 PLD 上に実装し，その他のデータバッファやフラグ用 D-FF をディスクリートで構成していく方法は，FIFO の段数が増えていくにしたがって現実的でなくなる．従って，小規模 PLD よりも容量が 10~20 倍程度で，入出力ピン数も 60 本程度とることができる FPGA を採用した．

(2) カスタムゲートアレイに比較して開発サイクルが短く開発コストが低い

大学の研究室等では，その性格上，ユーザの手許でハードウェアへ実装してその有効性等を確かめる作業がくり返し行われ，しかもそのサイクルは短い．従って，メーカへの依頼後からの開発期間が 2~3 週間程度のカスタムゲートアレイでは，開発サイクルが長すぎ，また開発コストも高くなる．実装後の不具合いを修正し

て，再び開発を依頼する，といった開発サイクルでは非常に効率が悪い．それに対して，FPGA は開発に必要な環境への投資が，比較的安価で済み，ユーザの手許でデバイスの作成やシミュレーション等を行っての動作確認や設計の修正が可能である．また，デバイスは 1 個単位で書き込みを行えるなど，開発コストが低く抑えられる．更には，FPGA のデバイスの供給が，複数のベンダーによって行われていることで，将来にわたってのデバイス供給体制の信頼度が高い．従って，小規模 PLD と同様な開発環境を整えることができる FPGA を採用した．

A.2 FPGA への実装手順

FPGA への実装は，次の手順 (1)~(4) で行った．作成に要した時間は，この試作が FPGA 設計ツールを用いる最初だったこともあり，手順 (1)~(3) の部分である回路設計に 7 日，手順 (4) の部分である設計の検証やシミュレーションに 5 日程度の時間を要した．この開発サイクルに要する時間は，ツールの仕様の慣れにしたがって，ある程度の短縮が見込まれる．

手順 (1) 待ち行列管理用セルのステートマシンを PLD 用ソースで記述

まず，**1.4** で説明した，待ち行列管理用セルのステートマシン（図 1.9）の記述は，PALASM2 [10] の PLD 用ソースファイルの形式で行うこととした．これは，以前から我々はこの形式でのステートマシン記述と，実際の PLD への実装の経験を積んでいたことで，そのためにプラットホームや開発ツールの準備はできていたことによる．また，次の (2) で，このステートマシンを回路設計ツール上で利用できるように変換するツールは，この PALASM2 のファイル形式をサポートしていたことにも，理由がある．セルのステートマシンを PALASM2 のソースファイルで記述したものを，図 A.1 に示す．

```
;PALASM2 Design Description
;---------- Declaration Segment ------------
TITLE    Cell State Machine Specification for FIFO
PATTERN  BCTRL1.PDS
REVISION 1.0
AUTHOR   K.Wasaki and N.Wakimoto
COMPANY  FACULTY OF ENGINEERING SHINSHU UNIVERSITY
DATE     Jan 13,1993
CHIP     BCTRL1 PAL22V10
;---------- PIN Declarations ---------------
CLK  RST  PR   FF1  FF2  NC  NC  NC  NC  NC  NC  GND
NC   NC   NC   NC   NC   NC  NC  NC  NC  Q0  G1  VCC
;--------- Boolean Equation Segment --------
EQUATIONS
/G1 :=                                  * /RST
     +  G1 *  Q0 * /PR *  FF1 * /FF2 *  RST
     + /G1 * /Q0 * /PR                  *  RST

/Q0 :=                                    /RST
     +  G1 *  Q0 * /PR                  *  RST
     +       /Q0 * /PR                  *  RST
;-----------------------------------------
```

図 A.1: セルを PALASM2 のソースファイルで記述

Fig A.1: PALASM2 specification file for a cell.

手順 (2) PLD 用ソースファイルを，回路設計ツール用のソフトマクロに変換

手順 (1) で記述した，管理用セルのステートマシンを，実際に設計を行うための回路設計用ツール（WORKview [18]）で利用するためには，ツール上でのモジュールに変換するための自動変換ツール（ALES1 [13]）を使用して，これをソフトマクロ（=機能部品）に変換する．

この変換ツールを用いて (1) で記述したセルをソフトマクロに変換し，更にセルが制御するフラグおよびデータバッファを設計した時点での，FIFO メモリ一段分の基本回路を，図 A.2に示す．

手順 (3) 回路設計ツールで，FIFO メモリ全体をデザイン

手順 (2) で設計した基本回路を多段に接続し，FIFO とした．また，読みだし時にセルが出力するデータを取りだすバスは，FPGA が内部にハイインピーダンス状態を作りだすことができないことから，各セルからの出力をセレクタによって選択し，出力するようにした．このデータセレクタは，8-to-1bit セレクタのマクロを組み合わせることにより構成した．

回路設計ツールを用いて，データセレクタをデザインした回路を，図 A.3に示す．

更に，メモリに格納されているデータの数を外部へ出力するためには，各セルが制御しているフラグの状態をエンコードする *FlagLogic* を配置した．この *FlagLogic* は，先の手順 (1)(2) による方法を用いて PALASM2 で記述したエンコーダを変換し，設計ツール上でのマクロとして配置した．

回路設計ツールを用いて，FIFO メモリ全体をデザインした回路を，図 A.4に示す．

手順 (4) ツールを用いての設計検証と，シミュレーション

試作には，テキサスインスツルメンツ社の FPGA [15] を用いた．実際に試作を行った FPGA は回路規模が約 2000 ゲート換算程度の比較的小規模な回路を対象としているものである．

この FPGA を製作するための設計検証とタイミング解析を行うためのツールは TI-

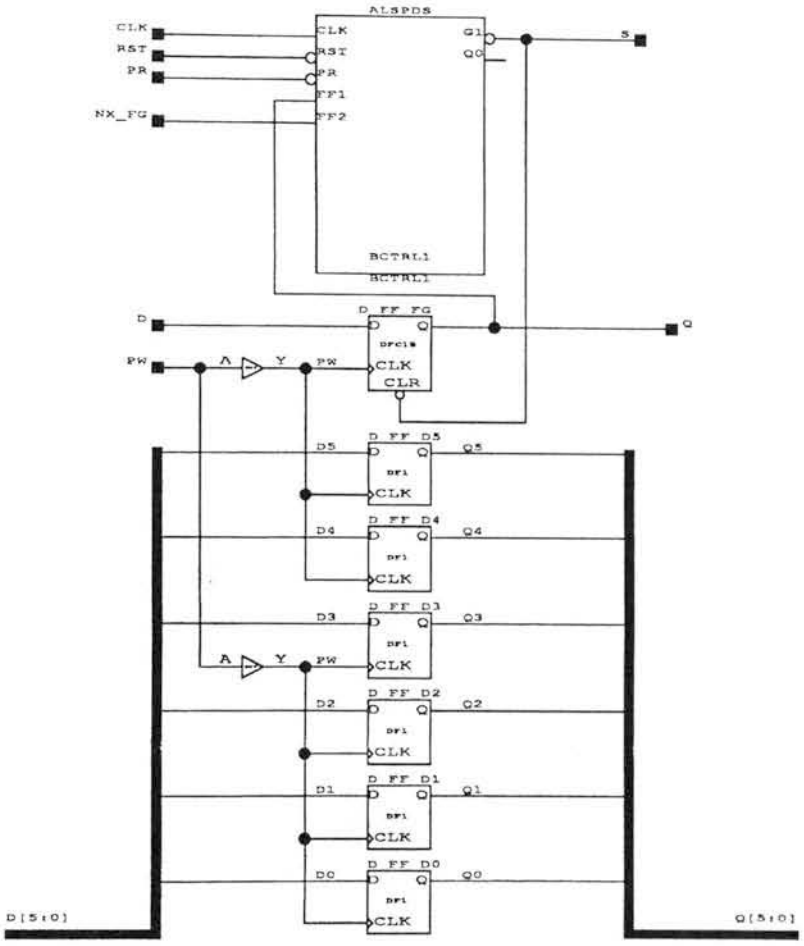


図 A.2: FIFO メモリー段分の基本回路

Fig A.2: Basic schematic of FIFO memory.

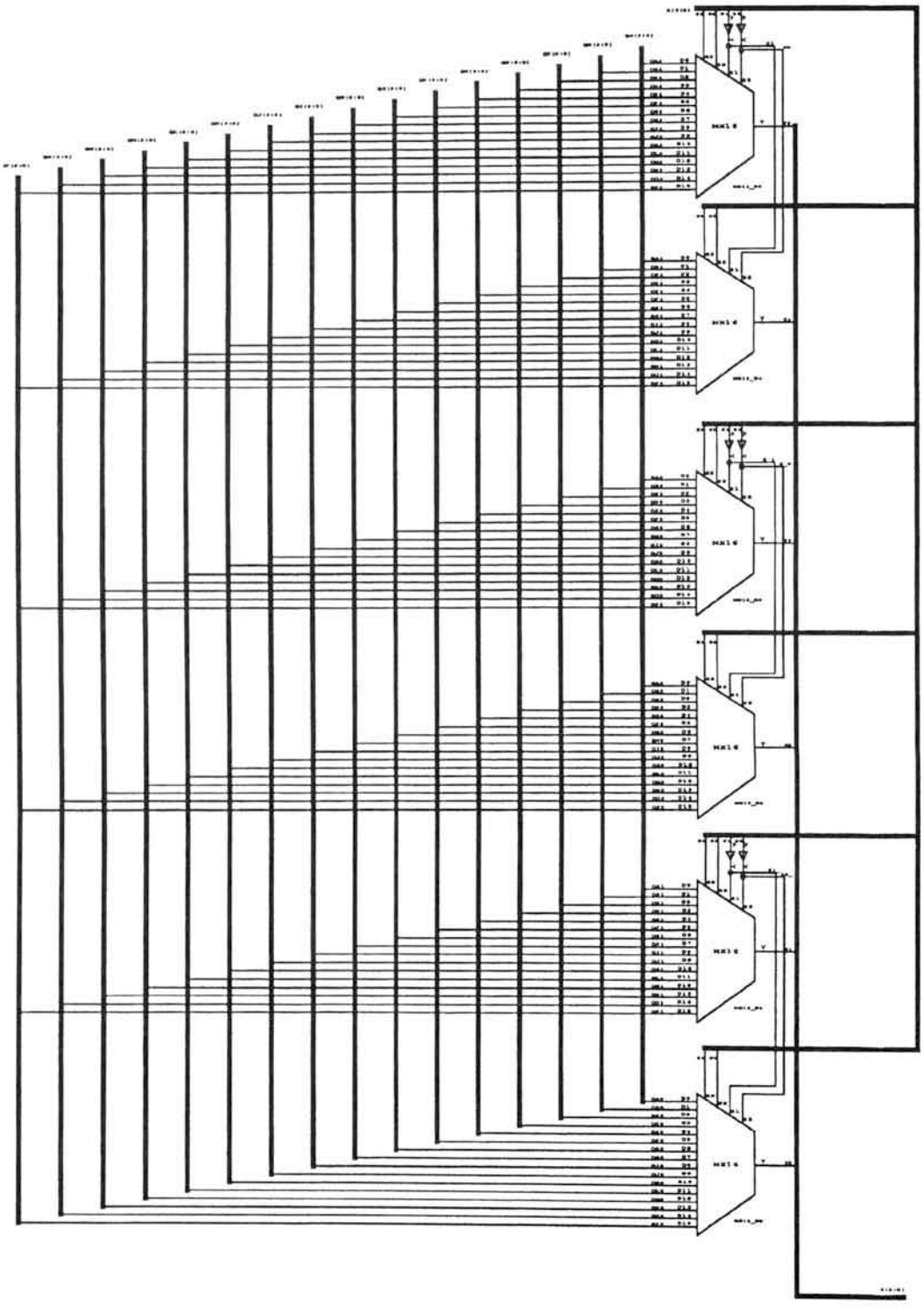


図 A.3: データセレクタの回路

Fig A.3: Schematic design of Data Selecter.

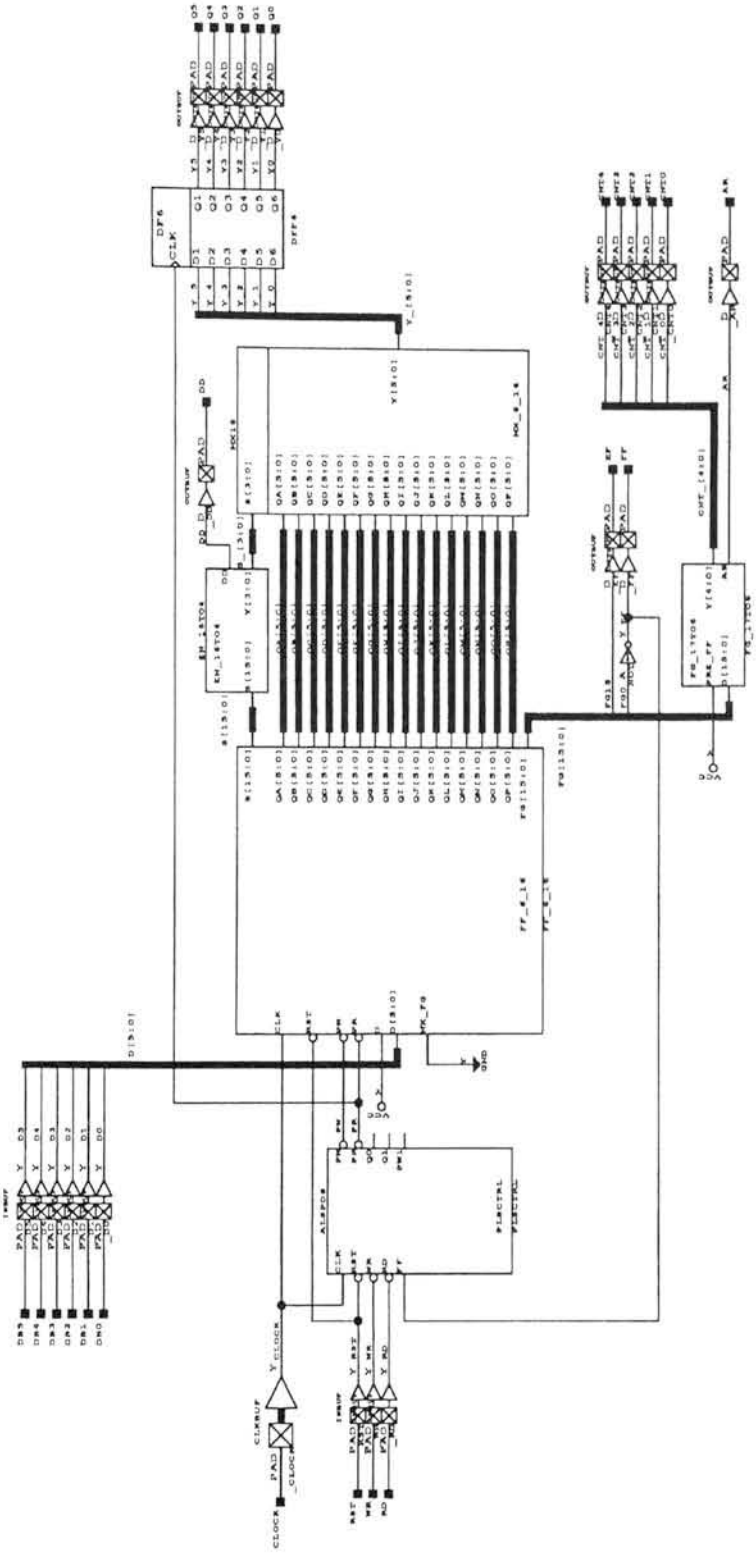


図 A.4: FIFO メモリ全体の回路

Fig A.4: Schematic design of the proposed FIFO memory.

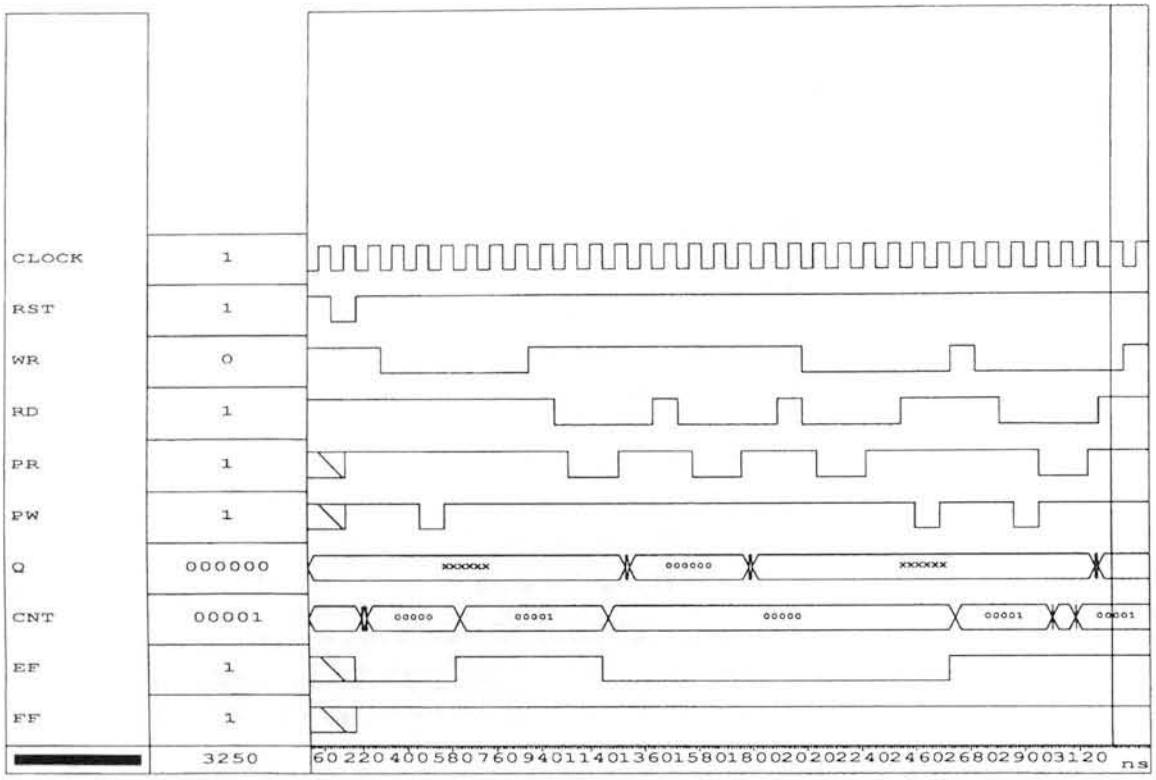


図 A.5: シミュレーションを行っている様子

Fig A.5: FIFO memory simulation .

ALS [14] で，このツール上で，本 FIFO メモリを FPGA に実装するための設計検証，および動作遅延時間などの解析を行った．ここで，TI-ALS 上で設計検証を行い，デバイスの内部遅延などタイミング情報を含めた条件下での動作確認を行っている様子を，図 A.5に示す．

実際のスイッチング特性においては，読みだし操作時のアクセスタイムが40nsec，書込み・読みだし操作後からのフラグ出力遅延が最大 160nsec 程度で十分に実用可能な値となった．また，この時間は，対象としたFPGA 自体の最大動作周波数の枠内で設計・試作を行った結果であり，今後，高速なFPGA デバイスの登場に従い，更に短縮が見込まれる．

A.3 FPGA 使用の利点，問題点および様々な工夫

本 FIFO メモリを FPGA 上に実装することにより，小規模 PLD 上でのそれでは得られない，多くの利点を享受した．一方，FPGA の機能が実装前の予想とは異なっていた部分では，設計前に用意した機能ブロックを冗長にしたり，回路自体を変更して対応したりした場面もあった．以下に，FPGA 使用の利点，問題点および様々な工夫についてまとめる．

まず，FPGA 使用に伴って享受した利点について挙げる．

(1) 初めて開発ツールを使用したにも関らず，設計開始からデバイス書き込みまでの開発サイクルが 2 週間弱であり，短かかった．これは，各ツールのマンマシンインターフェースが優れていること等が挙げられる．

(2) 開発ツールを稼動させるプラットホームがワークステーションだけでなく，パーソナルコンピュータも使用できるため，FPGA 開発環境と整えるための初期投資が必要最小限で済んだ．

(3) セルのステートマシンは PLD 用のソースファイルの形式で記述したが，それを FPGA 上のマクロとして登録し使用できる自動変換ツールが用意されていたために，管理セル部分の実装は容易であった．

(4) ツール上でメモリ動作の検証を行ったり，故意に待ち行列管理部分に故障を発生させて，自己回復性の確認ができた．また，デバイスへの書き込み後においても，数本用意されている外部テストピンにより，デバイス内部の様子が観察できたり，デバイスに故意に故障をひき起こすことができた．

(5) 設計検証とタイミング検証後は，実際の遅延情報を含めた回路のシミュレーションが可能であった．

一方，問題点および様々な工夫について挙げる．

(1) 2000 ゲートの容量をもつ FPGA に対して，ゲート数の限界近くまで回路を詰めこむと，設計検証の段階でつまづいてしまい，また可能であっても動作速度の点で問題が残る場合が多かった．デバイスが有しているゲート数と，実際に設計可能なゲート数とは若干の相違がある．

(2) 内部にハイインピーダンス状態を持たせることができないので，出力バスの部分はセレクタに変更しなければならかなった．この変更に伴ってセレクタを制御する回路等を付加したため，この冗長な部分がゲートを消費し，FIFO メモリの段数を減少させる原因になった．

(3) PLD では設計や実装を行う前から，出力遅延時間や入力に関するセットアップ時間などは決まっていて，その周辺回路の見通しがはっきりしている．しかし，FPGA を用いた場合，回路を設計し，設計検証とタイミング検証を行った後でないと，デバイスの外部から見た入出力タイミング情報は得られない．

(4) 今回使用した FPGA は，一回書きこむと消去できないタイプであるので，設計を少しだけ変更して再び実装する，といった実験のくり返しには不向きである．

付録 B　BMN から回路への変換

本論文で提案した LC-Petri net 等, ハイレベルペトリネットをそのままハードウェア化すると，回路が複雑になり過ぎる．ハードウェア化に適したペトリネット (Boolean Marking Net : BMN) は既に定義されている [26]．BMN は機械的に回路に変換できる．

B.1　Boolean Marking Net (BMN)

BMN の定義は以下の通りである.BMN は，従来のペトリネットに対し，ハードウェア化が容易な様にプレースに収容するマークについて変更したものである．このような制限を加えても，現実のシステムの解析・設計には十分有効であることがわかっている [25].

定義 B.1 (*Boolean Marking Net*)

Boolean Marking Net $BMN = (S, T; F)$ は次の各条件を満たす

(i) $S = \{s_1, s_2, \cdots, s_n\}$ 及び $T = \{t_1, t_2, \cdots, t_m\}$ は，それぞれプレース，トランジションの集合であり，$F \subseteq (S \times T) \cup (T \times S)$ はプレースからトランジション，もしくはトランジションからプレースへのアークの集合である．

(ii) 各プレース $s_i (\in S)$ のマークの容量は 1.

(iii) 各アーク $f_i (\in F)$ の重みは 1.

(iv) 各プレース s_iのマークの個数は，0 か 1．このため，ネット全体のプレースのあらゆるマーク状態からなる集合 Pは $P = 2^S$で表せる．

(v) $t_i(\in T)$ へのアークを有する全てのプレースの集合を${}^*t_i$，t_iからの全てのアークが指すプレースの集合を$t_i{}^*$と書く．つまり，

$$
\begin{aligned}
{}^*t_i &= \{s \in S : ({}^\exists f)(f \in F, f = (s, t_i))\} \\
t_i{}^* &= \{s \in S : ({}^\exists f)(f \in F, f = (t_i, s))\}
\end{aligned}
$$

t_iに関する**発火評価**とは，${}^*t_i$に属する全てのプレースがマークを有しているか否かを調べ，有していればこれらのプレースからマークを取り，t_i^*に属する全てのプレースのマークの個数を 1 にすることである．

このトランジションt_iの発火評価は，以下に示すネットのプレースのマーク状態$M \in P$から$\varphi_i(M) \in P$への写像で表すことが出来る．

$$
\varphi_i(M) = \begin{cases} 0 & : on \ \ {}^*t_j - t_j{}^* \\ 1 & : on \ \ t_j{}^* \\ M & : otherwise \end{cases}
$$

この発火評価の特徴は，発火時のプレースへのマークの書き込み方にある．従来のペトリネットでは，発火するとt_i^*に属する全てのプレース内のマークを 1 増やす．しかし，これをそのままハードウェア化しようとすると，各プレースはカウンタで構成しなければならず，ハードウェアの規模が大きくなってしまう．このため，ここでは$t_i{}^*$に属するプレースのマークの数を 1 にすることにし，各プレースを 1 つの flip-flop で構成できるようにした．このマークの置き方は，各プレースがマークの有無で論理を表しており，t_iが発火すると$t_i{}^*$に属するプレースに Boole 代数的にマークを加算すると見なすことが出来ることから，このペトリネットを Boolean Marking Net と呼ぶ． □

ここで簡単な BMN の例を図 B.1に示す．このネットは，最初にプレースs_3, s_4にマークを置き，プレースs_5にマークが無い状態にしておき，トランジションの発火評価をt_2,t_1,t_3,t_4の順で行う．すると，最初プレースs_1,s_2の両方ともにマークが有るか，両方ともマークが無い状態であればt_4発火評価直後s_5にマークが有る状態となり，s_1,s_2の状態がそれ以外ではs_5にマークが無い状態となる．つまり，この BMN はs_1,s_2の比較器として動作する．

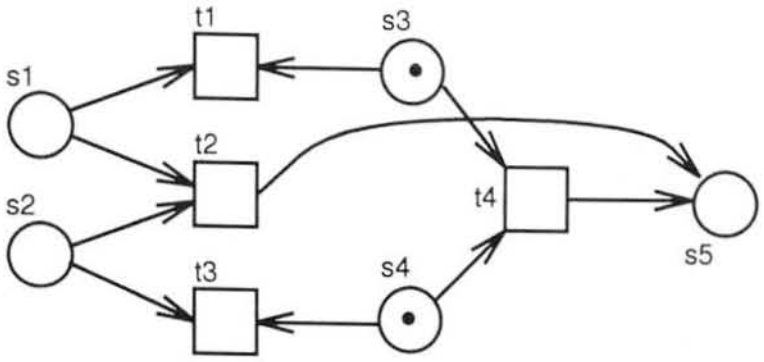


図 B.1: 比較器を BMN でモデル化した例

FigB.1: An Example of BMN for a Comparator.

B.2 BMN から回路への変換手順

B.2.1 ハードウェア化の手法

BMN のハードウェア化は，以下のように機械的に行う．図 B.2にトランジション t_i の周辺回路を示す．

各プレースはマークの有無を保存するため RS-FF に対応させる．マークがある状態は，RS-FF がセット (Q=1) されていることで表す．

各トランジションは発火評価のタイミングが問題となるため，クロックで制御することを考える．そのため，各トランジションには，Positive Edge Trigger 型の D-FF(D+) と，Negative Edge Trigger 型の D-FF(D−) の対を対応させる．

トランジション t_iの発火評価は，クロック c_iの Positive Edge で${}^{\bullet}t_i$に属する全てのプレースにマークが有るかどうかを調べて，発火するか否かを決定する．発火する場合は，${}^{\bullet}t_i$に属する全てのプレースからマークを取る．t_iが発火した場合は，その後 c_iの Negative Edge で，$t_i^{\bullet}$に属する全てのプレースにマークを置く．また，図中 c_jは，t_iの次に発火評価されるトランジション t_jの発火評価タイミング用信号である．

トランジション t_iは，発火評価される時に${}^{\bullet}t_i$に属する全てのプレースがマークを有する場合，発火する．このため，${}^{\bullet}t_i$に属するプレースのマークの有無を表す RS-FF の出力を全て AND 回路の入力端子に接続し，AND 回路の出力を D+の入力端子に入れる．

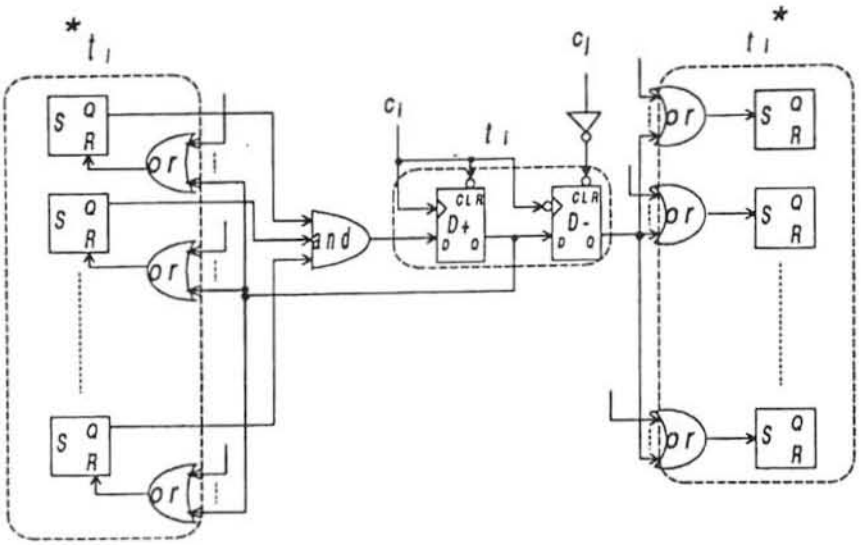


図 B.2: トランジション t_iの周辺回路

FigB.2: A Hardware Implementation for the peripheral circuits of t_i.

トランジション t_iが発火して${}^*t_i$に属する各プレースからマークが除去されることは，すなわち D+の出力をそれらのプレースを表す RS-FF のリセット端子に接続することで表される．${}^*t_i$に属する各プレースが，他のトランジション t_kの${}^*t_k$にも属している場合には，t_kの D+からのリセット信号も接続されるため，各トランジションからのリセット信号は，OR 回路で取りまとめた上で，OR 回路の出力をプレース RS-FF のリセット端子へ接続する．

一方，t_i^*に属するプレースにマークが置かれることは，すなわち D-の出力をそれらのプレースを表す RS-FF のセット端子に接続することで表される．複数のトランジションからアークが伸びているプレースでは，アークを伸ばしているトランジションのうち，どれか一つでも発火するとマークが置かれるため，各トランジションの D-の出力は，OR 回路で取りまとめた上で，OR 回路の出力をプレースの RS-FF のセット端子に接続する．

t_iの処理に関して，発火した場合の${}^*t_i$からのマークの取り除き処理と，t_i^*へのマークの書き込み処理とは，${}^*t_i \cap t_i^* \neq \phi$である場合があるため，同時には行うことができない． BMN の定義で示した$\varphi_i(M)$ より，${}^*t_i \cap t_i^*$に属するプレースには，発火後マークが置かれていなければならない．このため，マークの取り除き処理と書き込み処理を

D+とD−の対を用いてc_iの立ち上がりと立ち下がりに分けて行うことで，1クロックで当該トランジションの評価が完了できるようにした．

図B.3は図B.1のBMNを上記の手順でハードウェア化したものである．

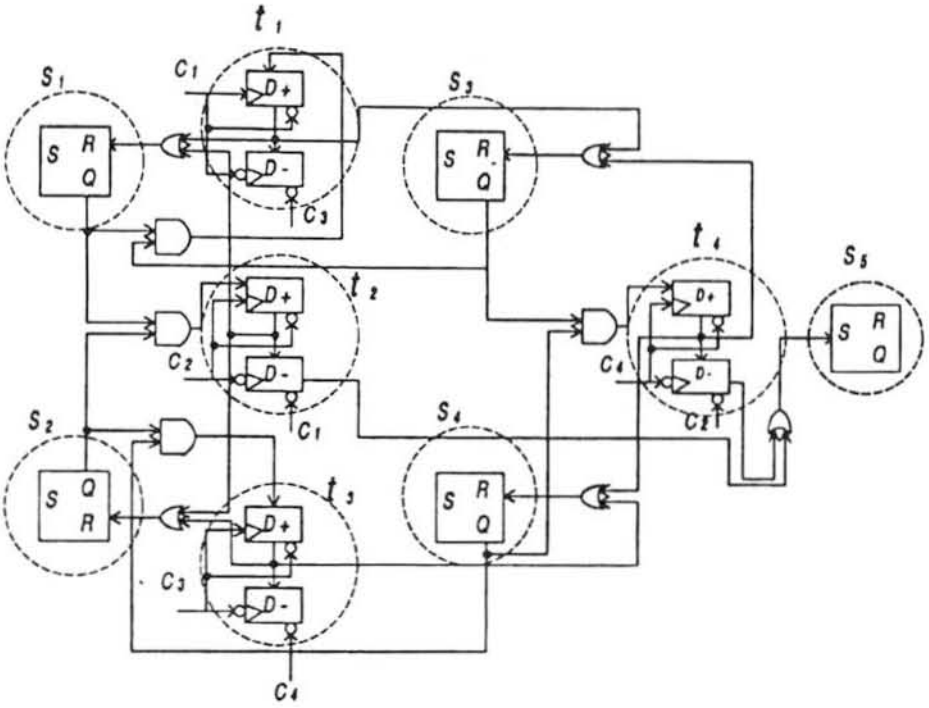


図 B.3: BMNから回路へ変換された例

FigB.3: An Example of Hardware Implementation from the BMN.

B.2.2 駆動クロックの与え方

設計者は，上記で示したハードウェア化手法で作成した回路を動作させるために，各トランジションを発火評価するクロックを与えなければならない．ここで，作成した回路が目的に合った動作をするためには，各トランジションをどのような順序で発火評価するかを決め，この順番でクロックを与えなければならない．

先の図B.1で示したシステムは，t_2, t_1, t_3, t_4 の順で発火評価すると正しく動作する．このため，図B.3で示した回路において，各トランジションを示すD-FFに与えるクロック c_1, c_2, c_3, c_4 は，図B.4で示すように与える必要がある．この場合，順番に発火

評価するためにタイミングの異なる4個のクロックが必要である.

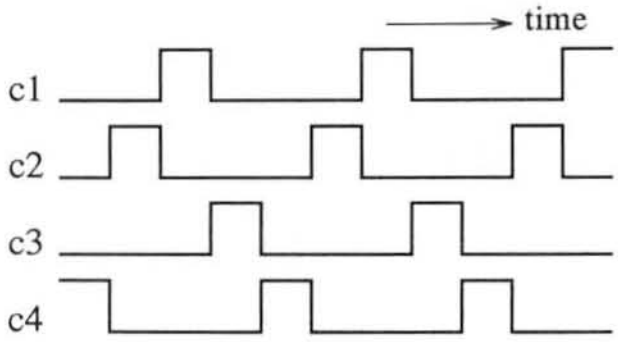


図 B.4: 位相をずらした動作クロック

FigB.4: Phase-shifted Clocks for the Comparator Circuit.

このハードウェアにおいて，ある複数のトランジションの発火評価を同時に並行して行っても結果がもとの評価順序の場合と同じであれば，そのトランジションに与えるクロックを同一にして発火評価を並行させることで，クロック数を減少でき，回路を高速化できる.

高速化の検討のためには，与えられた正しい動作を行うことが分かっているトランジションの発火評価順序と，等価な動作を行う異なるトランジションの発火評価順序を全て求める．ここで，等価とは何かを明らかにする必要がある．次に，求めた様々な発火評価順序から，発火評価の順番を交換しても動作が等価であるトランジションの組を求める．この組のトランジションは，同時に評価してもやはり動作が等価となることを明らかにする必要がある．トランジション発火評価順序の等価性，および交換可能性については既に検討が行われている [29][24][25][26] [27]．これらの組（トランジション・グループ）に対して同位相のクロックを与えることにより，高速化した回路を作成する.

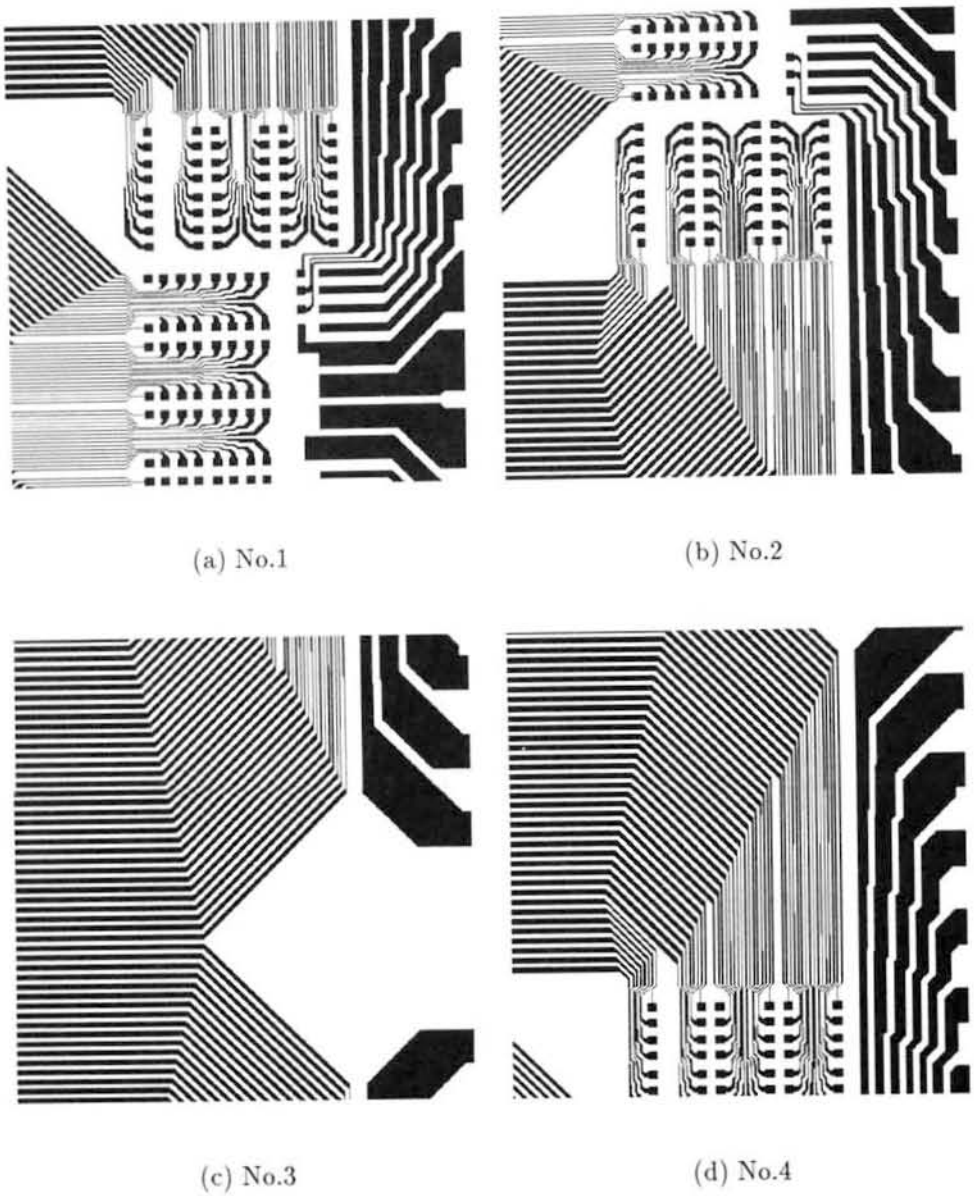
付録 C　PCB 画像と CCITT テストチャートの Run-length 分布

C.1　各種符号法による圧縮率比較

表 C.1: テスト画像に関する圧縮率 (%) の比較

	Run-Length	MH	BPC
PCB–1	17.5	27.3	16.9
PCB–2	23.9	15.5	21.0
PCB–3	11.8	20.6	11.8
PCB–4	23.3	17.4	22.2
Average	19.1	20.2	18.0
CCITT–1	6.2	9.8	9.1
CCITT–2	5.7	7.8	9.3
CCITT–3	11.9	13.5	17.6
CCITT–4	21.3	22.4	30.3
CCITT–5	12.7	14.1	18.3
CCITT–6	9.5	11.0	14.5
CCITT–7	22.3	21.5	33.3
CCITT–8	11.6	12.8	18.4
Average	12.7	14.1	18.9

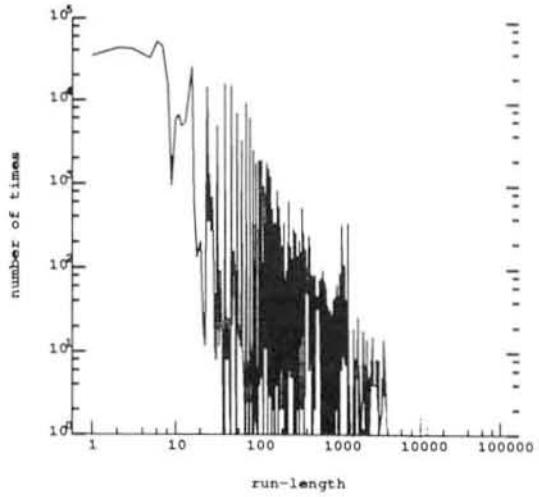
C.2 各種画像例と Run-length 分布



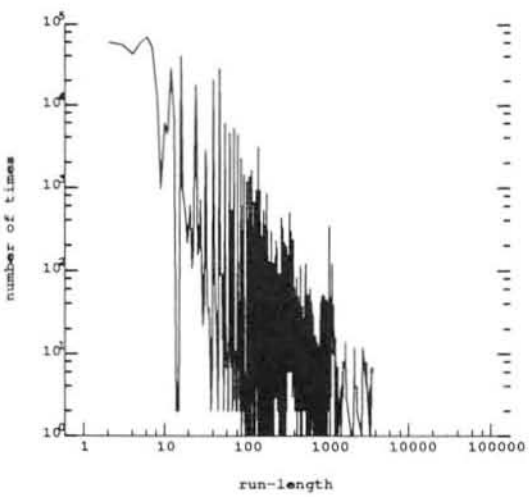
(a) No.1　(b) No.2

(c) No.3　(d) No.4

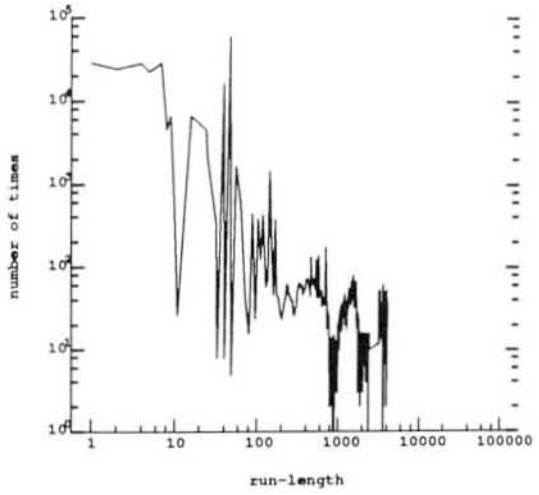
図 C.1: PCB 配線パターン画像 No.1〜4 (4096×4096 画素)



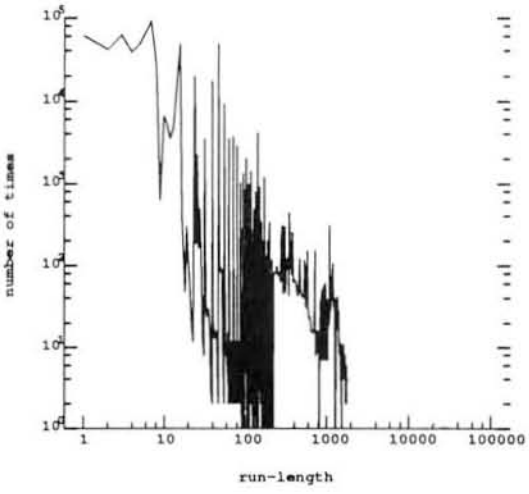
(a) No.1



(b) No.2

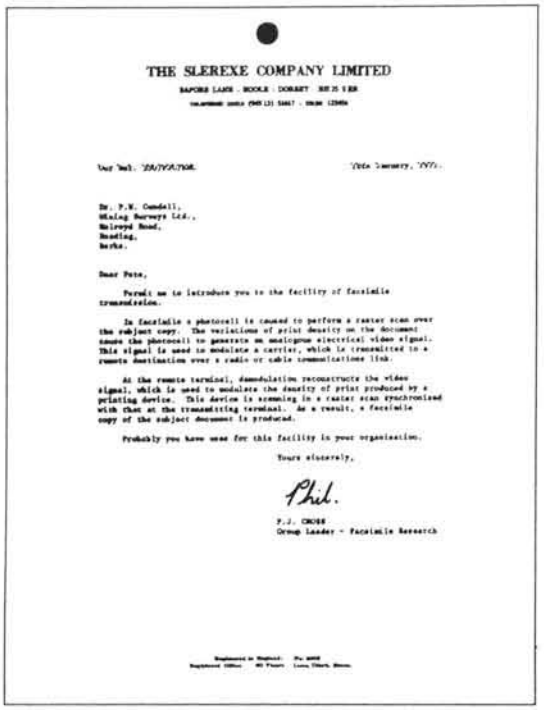


(c) No.3



(d) No.4

図 C.2: PCB 配線パターン画像 No.1〜4 の Run-length 分布



THE SLEREXE COMPANY LIMITED

Dr. P.K. Cundell,
Analog Surveys Ltd.,
Bulroyd Road,
Reading,
Berks.

Dear Pete,

Permit me to introduce you to the facility of facsimile transmission.

In facsimile a photocell is caused to perform a raster scan over the subject copy. The variations of print density on the document cause the photocell to generate an analogous electrical video signal. This signal is used to modulate a carrier, which is transmitted to a remote destination over a radio or cable communications link.

At the remote terminal, demodulation reconstructs the video signal, which is used to modulate the density of print produced by a printing device. This device is scanning in a raster scan synchronised with that at the transmitting terminal. As a result, a facsimile copy of the subject document is produced.

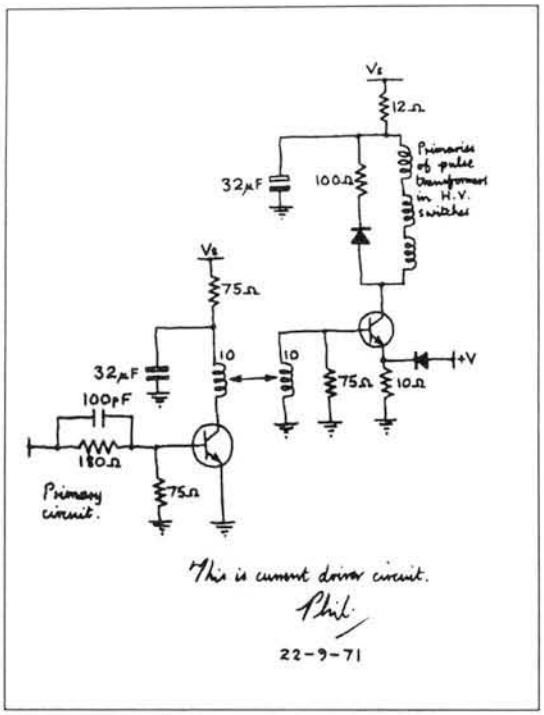
Probably you have uses for this facility in your organisation.

Yours sincerely,

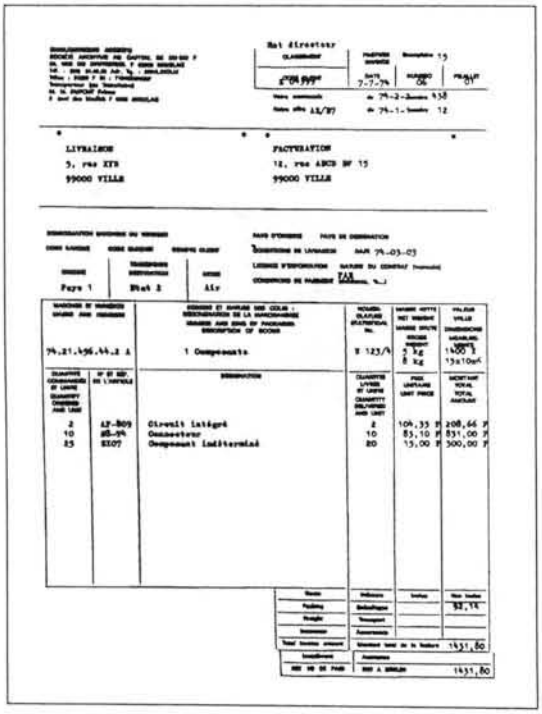
Phil.

P.J. CROSS
Group Leader - Facsimile Research

(a) No.1



(b) No.2



Chez directeur

LIVRAISON
5, rue XYZ
99000 VILLE

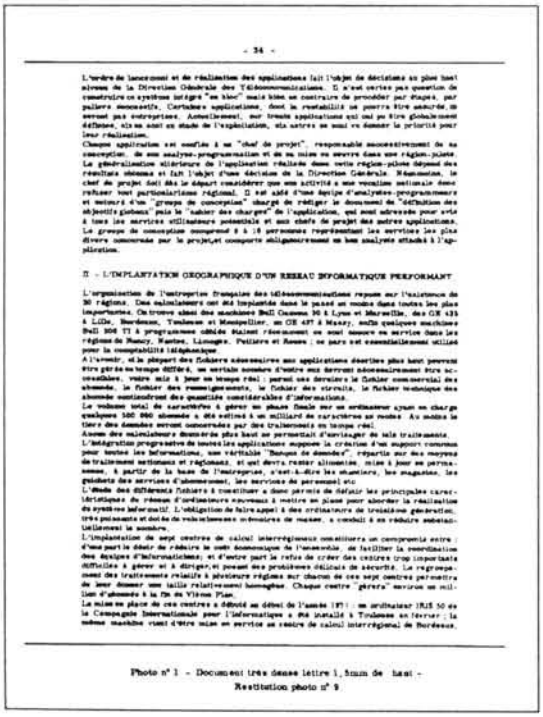
FACTURATION
12, rue ABCD BP 15
99000 VILLE

Pays 1 | Mod 2 | Air

2	47-809	Circuit intégré	104,35 F	208,66 F
10	85-94	Connecteur	83,10 F	831,00 F
25	5107	Composant indifférentiel	20	15,00 F 500,00 F

1431,80

(c) No.3



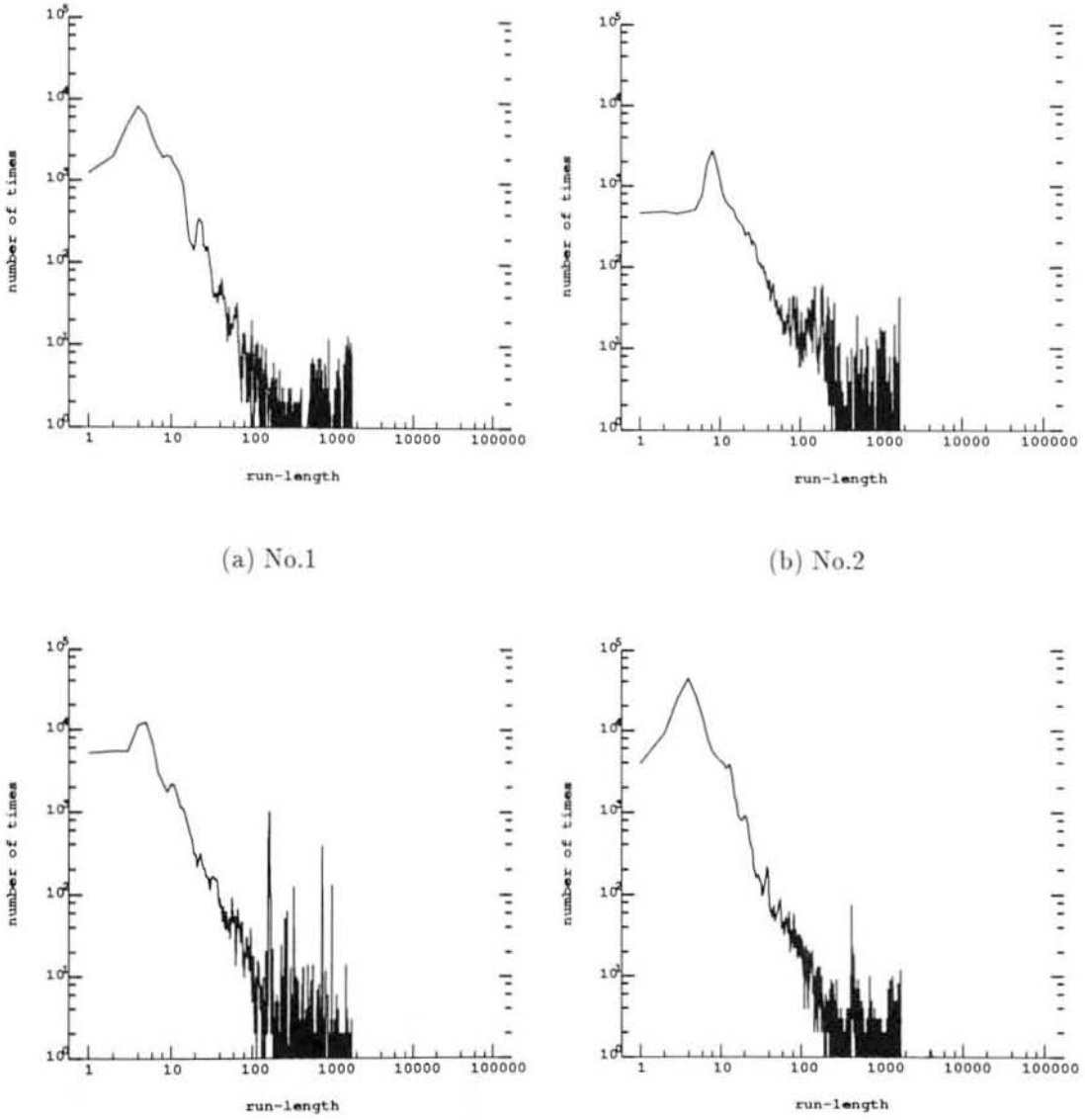
- 34 -

II - L'IMPLANTATION GEOGRAPHIQUE D'UN RESEAU INFORMATIQUE PERFORMANT

Photo n° 1 - Document très dense lettre 1,5mm de haut -
Restitution photo n° 9

(d) No.4

図 C.3: CCITT テストチャート No.1〜4 (1728×2376 画素)



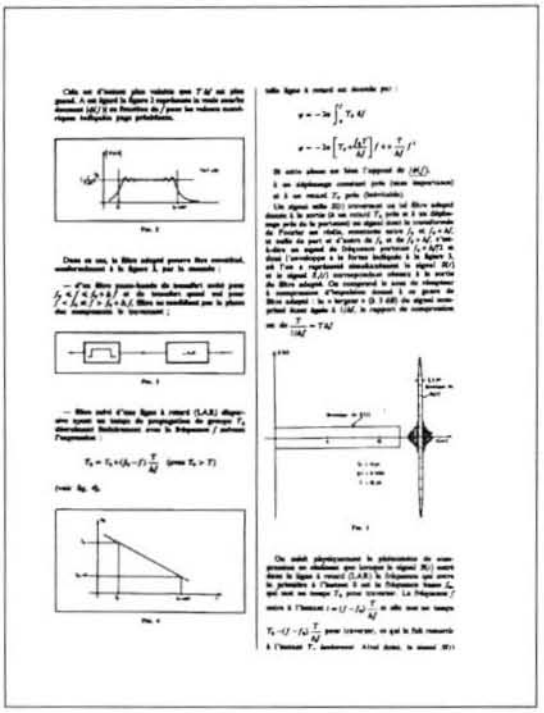
(a) No.1

(b) No.2

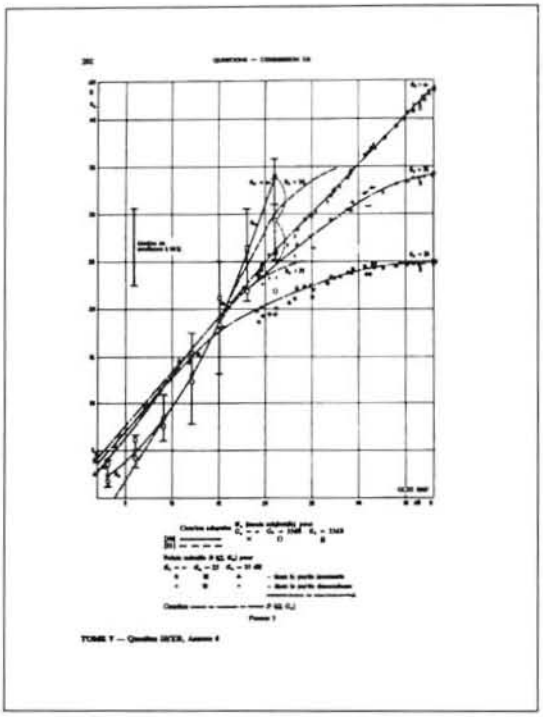
(c) No.3

(d) No.4

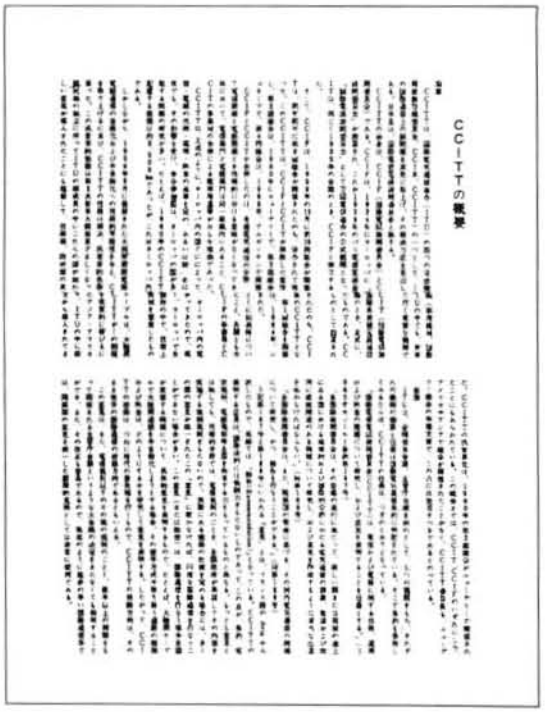
図 C.4: CCITT テストチャート No.1〜4 の Run-length 分布



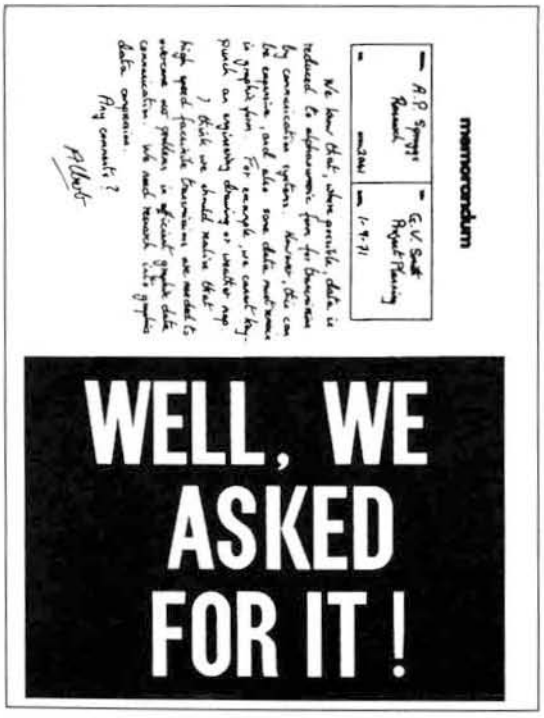
(a) No.5



(b) No.6

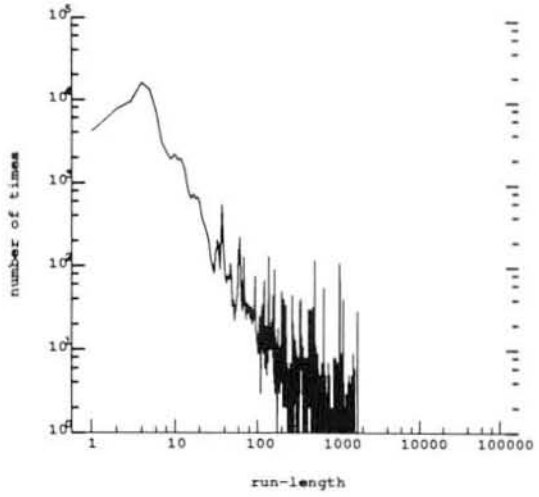


(c) No.7

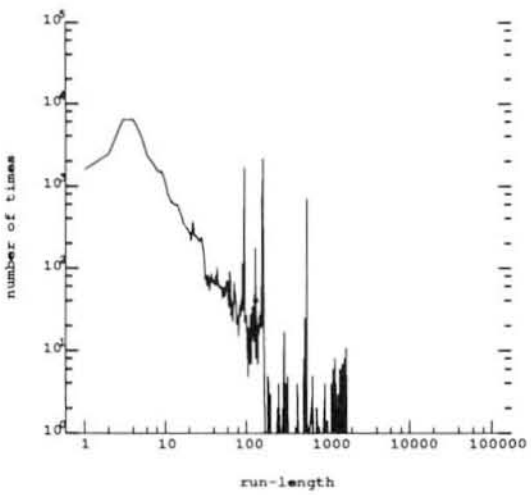


(d) No.8

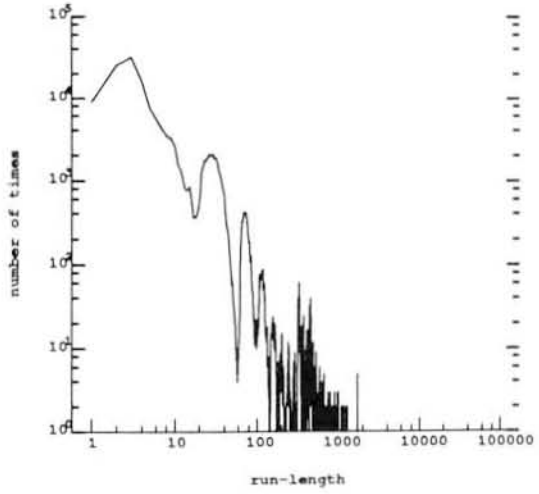
図 C.5: CCITT テストチャート No.5〜8 (1728×2376 画素)



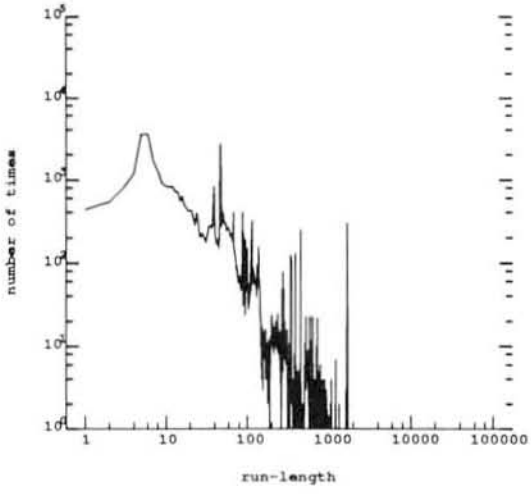
(a) No.5



(b) No.6



(c) No.7



(d) No.8

図 C.6: CCITT テストチャート No.5〜8 の Run-length 分布

付録D　2の補数回路に関する *Mizar article*

D.1　*Mizar*による形式化

Mizar[70][71]は, Warsaw大学のA.Trybulecらを中心とした, *Mizar Association*[1]によって進められている, 数学を計算機を使って形式化するプロジェクトの総称である. このプロジェクトの目的は, 数学の論文の検査システムを作ることである. *Mizar*プロジェクトは, 計算機を使って数学を記述するために作られた*Mizar*言語の文法[70]に則った記法で命題の証明を記述し, それを計算機上で動作する*Mizar Proof Checker*によって証明の各ステップの正当性を検査し, 証明が検査済みの命題をライブラリ化[71]することによって行われている.

*Mizar*によって数学の証明を形式化し記述した論文を*article*と呼ぶ（例えば文献[72]···[105]）. 新たに*article*を記述する際には, 過去既に証明が検査済みでライブラリに登録されている沢山の*article*を参照しながら進める. 記述した*article*がライブラリに登録された後, その*article*は他の*article*が参照することができる. *article*は*Mizar*言語によって記述する. *Mizar*言語は, 数学の一般的な証明の記述法を基にしているが, *Mizar*独特の記法もあるので, 文法の詳細に関しては文献[70]を参照されたい.

[1] *Mizar Users Association* : Institute of Mathematics Warsaw University, Bialystok Branch ul. Akademicka 2, 15-267 Bialystok, Poland.

D.2 項と式の記法

D.2.1 項（*terms*）

(1) 変数識別子 (*identifiers of variables*) は項である.

（例） $p,\ f,\ g,\ x_1,\ x_2,\ o,\ S,\ S_1,\ S_2,\ s,\ v,\ G$

(2) 数値 (*numerals*) は項である.

（例） $0,\ 1,\ 2,\ \cdots,\ 100,\ \cdots$

(3) 関数 (*functor*) を含む以下の形式は項である.

$leftside - arguments \quad functor - symbol \quad rightside - arguments$

（例） $\emptyset,\ P \cup U,\ P \cap U,\ \neg x_1,\ x_1 \wedge x_2,\ and2a(\{x_1, x_2\}),\ Arity(o)$

(4) 空でない項を括弧 (*bracket*) で囲んだ以下の形式は項である.

$leftside - bracket \quad non - zero - terms \quad rightside - bracket$

（例） $[: M1, M2 :]\,(Cartesian\ product),\ \langle * x_1, x_2 * \rangle\,(finite\ sequence),\ \{x, y\}\,(finite\ s$

(5) 項の選択子 (*selector*) を含む以下の形式は項である.

$the \quad symbol - of - selector \quad of \quad term$

（例） $the\ carrier\ of\ S,\ the\ topology\ of G$

(6) 項の構造型 (*structure*) を含む以下の形式は項である.

$symbol - of - structure \ \ll \ list - of - terms \ \gg$

（例） $ManySortedSignature \ll carrier,\ operation\ symbols,\ arity,\ result\ sort \gg$

D.2.2 型（*types*）

(1) 以下の様なクラスは *mode* という.

（*mode* の例） $Any,\ set,\ Element\ of\ X,\ DOMAIN,\ Subset\ of\ D,\ Real,\ Nat,\ Boole$

ここで, *mode* を含む以下の形式は型である.

$symbol - of - mode \ \ of \ \ terms$

（例） $Subset\ of\ G$, $Subset\ of\ the\ carrier\ of\ G$, $Relation\ of\ X$

(2) 構造型識別子 (*symbol-of-structure*) は型である.

（例） $ManySortedSignature$, $TopStruct$, $LattStr$

D.2.3 式 (*formulas*)

(1) 述語式 (*predicative formula*) は式である.

$terms \ \ symbol - of - predicate \ \ terms$

$terms \ \ is \ \ type$

（例） $InnerVertices\ CompStr(x,b)\ is\ Relation$

$InputVertices\ CompStr(x,b)\ is\ without - pairs$

(2) 述語命題 (*proposition formula*) は式である.

$for \ \ identifiers - of - variables \ \ holds \ \ formula$

$for \ \ segment - of - qualified - variables \ \ holds \ \ formula$

（例） $for\ x, y\ being\ Element\ of\ BOOLEAN\ holds\ xor2b.\langle *x, y* \rangle = xor2.\langle *x, y* \rangle$

D.3 *2's Complement Circuit Mizar article*

```
:: 2's Complement Circuit.
::  by Katsumi Wasaki and Pauline N. Kawamoto
::

environ

 vocabulary PAIR,RELATION,PBOOLE,FUNC,FUNC_REL,FINSEQ,TUPLES,CIRCOP1,VALUAT,
     ZF_LANG,NISHIYA,LATTICES,BOOLE,MSUALG_1,FUNC4,AMI,AMI2,CIRCUITS,FULLADD,
     TWOSCOMP;

 constructors ARYTM,REAL_1,NAT_1,MCART_1,CARD_3,FINSET_1,DOMAIN_1,MARGREL1,
     BINARITH,CLASSES1,MSAFREE2,CIRCUIT1,CIRCUIT2,CIRCCOMB,ENUMSET1,PRVECT_1,
     FACIRC_1;

 requirements ARYTM;

 notation ARYTM,STRUCT_0,TARSKI,BOOLE,ENUMSET1,NAT_1,MCART_1,RELAT_1,FUNCT_1,
     FUNCT_2,REAL_1,INT_1,MARGREL1,BINOP_1,VECTSP_1,FINSEQ_1,FINSET_1,FINSEQ_2,
     FINSEQ_4,FUNCOP_1,FUNCT_4,BINARITH,ORDINAL1,CARD_3,CLASSES1,PARTFUN1,
     PRVECT_1,PBOOLE,MSUALG_1,PRE_CIRC,MSAFREE2,CIRCUIT1,CIRCUIT2,CIRCCOMB,
     FACIRC_1;

 clusters STRUCT_0,RELAT_1,FUNCT_1,INT_1,FINSET_1,RELSET_1,FINSEQ_2,FRAENKEL,
     PRVECT_1,PBOOLE,MARGREL1,MSUALG_1,PRE_CIRC,MSAFREE2,CIRCUIT1,CIRCCOMB,
     FACIRC_1;

 definitions TARSKI,BOOLE,RELAT_1,FINSEQ_1,MARGREL1,BINARITH,MSAFREE2,CIRCUIT2,
     CIRCCOMB,FACIRC_1;

 theorems TARSKI,BOOLE,ZFMISC_1,ENUMSET1,MARGREL1,BINARITH,FUNCT_1,FUNCT_2,
     FINSEQ_1,FINSEQ_2,FINSEQ_3,CIRCUIT1,CIRCUIT2,CIRCCOMB,FACIRC_1;

 schemes FACIRC_1;

begin

::-------------------------------------------------------------------------
:: Preliminaries
::-------------------------------------------------------------------------

definition
 redefine
 let x be set;
 func <*x*> -> FinSeqLen of 1;
  coherence proof thus len <*x*> = 1 by FINSEQ_1:57; end;
 let y be set;
 func <*x,y*> -> FinSeqLen of 2;
  coherence proof thus len <*x,y*> = 2 by FINSEQ_1:61; end;
 let z be set;
 func <*x,y,z*> -> FinSeqLen of 3;
  coherence proof thus len <*x,y,z*> = 3 by FINSEQ_1:62; end;
end;

definition let n,m be Nat;
 let p be FinSeqLen of n;
 let q be FinSeqLen of m;
 redefine func p^q -> FinSeqLen of n+m;
  coherence
   proof
    thus len (p^q) = len p + len q by FINSEQ_1:35
                  .= n+len q by CIRCCOMB:def 12
                  .= n+m by CIRCCOMB:def 12;
   end;
end;

definition let S be unsplit non void non empty ManySortedSign;
```

```
 let A be Boolean Circuit of S;
 let s be State of A;
 let v be Vertex of S;
 redefine func s.v -> Element of BOOLEAN;
  coherence
   proof s.v ∈ (the Sorts of A).v by CIRCUIT1:5;
    hence thesis;
   end;
end;

theorem PreLem2':
 for f being Function for x1,x2 being set st
  x1 ∈ dom f & x2 ∈ dom f holds
   f·<*x1,x2*> = <*f.x1,f.x2*>
 proof
  let f be Function;
  let x1,x2 be set; assume
A: x1 ∈ dom f & x2 ∈ dom f; then
   f.x1 ∈ rng f by FUNCT_1:12; then
  reconsider D = dom f, E = rng f as non empty set by A;
  reconsider g = f as Function of D,E by FUNCT_2:95;
   rng <*x1,x2*> = {x1,x2} by FINSEQ_3:35; then
   rng <*x1,x2*> c= D by A,ZFMISC_1:38; then
  reconsider p = <*x1,x2*> as FinSequence of D by FINSEQ_1:def 4;
  thus f·<*x1,x2*> = g·p .= <*f.x1,f.x2*> by FINSEQ_2:40;
 end;

theorem ::PreLem3':
 for f being Function for x1,x2,x3 being set st
  x1 ∈ dom f & x2 ∈ dom f & x3 ∈ dom f holds
   f·<*x1,x2,x3*> = <*f.x1,f.x2,f.x3*>
 proof
  let f be Function;
  let x1,x2,x3 be set; assume
A: x1 ∈ dom f & x2 ∈ dom f & x3 ∈ dom f; then
   f.x1 ∈ rng f by FUNCT_1:12; then
  reconsider D = dom f, E = rng f as non empty set by A;
  reconsider g = f as Function of D,E by FUNCT_2:95;
   rng <*x1,x2,x3*> = {x1,x2,x3} by FINSEQ_3:36; then
   rng <*x1,x2,x3*> = {x1,x2} U {x3} & {x1,x2} c= D & {x3} c= D
    by A,ZFMISC_1:37,38,ENUMSET1:43; then
   rng <*x1,x2,x3*> c= D by BOOLE:32; then
  reconsider p = <*x1,x2,x3*> as FinSequence of D by FINSEQ_1:def 4;
  thus f·<*x1,x2,x3*> = g·p .= <*f.x1,f.x2,f.x3*> by FINSEQ_2:41;
 end;

::-------------------------------------------------------------------------
:: Boolean Operations : 2,3-Input and*, nand*, or*, nor*, xor*
::-------------------------------------------------------------------------

:: 2-Input Operators

definition
  func and2 -> Function of 2-tuples_on BOOLEAN, BOOLEAN means:
AND: for x,y being Element of BOOLEAN holds it.<*x,y*> = x '&' y;
  correctness from 2AryBooleDef;
  func and2a -> Function of 2-tuples_on BOOLEAN, BOOLEAN means:
AND2A: for x,y being Element of BOOLEAN holds it.<*x,y*> = ¬x '&' y;
  correctness from 2AryBooleDef;
  func and2b -> Function of 2-tuples_on BOOLEAN, BOOLEAN means:
AND2B: for x,y being Element of BOOLEAN holds it.<*x,y*> = ¬x '&' ¬y;
  correctness from 2AryBooleDef;
end;

definition
  func nand2 -> Function of 2-tuples_on BOOLEAN, BOOLEAN means:
```

```
NAND: for x,y being Element of BOOLEAN holds it.<*x,y*> = ¬(x '&' y);
  correctness from 2AryBooleDef;
  func nand2a -> Function of 2-tuples_on BOOLEAN, BOOLEAN means:
NAND2A: for x,y being Element of BOOLEAN holds it.<*x,y*> = ¬(¬x '&' y);
  correctness from 2AryBooleDef;
  func nand2b -> Function of 2-tuples_on BOOLEAN, BOOLEAN means:
NAND2B: for x,y being Element of BOOLEAN holds it.<*x,y*> = ¬(¬x '&' ¬y);
  correctness from 2AryBooleDef;
end;

definition
  func or2 -> Function of 2-tuples_on BOOLEAN, BOOLEAN means:
OR: for x,y being Element of BOOLEAN holds it.<*x,y*> = x 'or' y;
  correctness from 2AryBooleDef;
  func or2a -> Function of 2-tuples_on BOOLEAN, BOOLEAN means:
OR2A: for x,y being Element of BOOLEAN holds it.<*x,y*> = ¬x 'or' y;
  correctness from 2AryBooleDef;
  func or2b -> Function of 2-tuples_on BOOLEAN, BOOLEAN means:
OR2B: for x,y being Element of BOOLEAN holds it.<*x,y*> = ¬x 'or' ¬y;
  correctness from 2AryBooleDef;
end;

definition
  func nor2 -> Function of 2-tuples_on BOOLEAN, BOOLEAN means:
NOR: for x,y being Element of BOOLEAN holds it.<*x,y*> = ¬(x 'or' y);
  correctness from 2AryBooleDef;
  func nor2a -> Function of 2-tuples_on BOOLEAN, BOOLEAN means:
NOR2A: for x,y being Element of BOOLEAN holds it.<*x,y*> = ¬(¬x 'or' y);
  correctness from 2AryBooleDef;
  func nor2b -> Function of 2-tuples_on BOOLEAN, BOOLEAN means:
NOR2B: for x,y being Element of BOOLEAN holds it.<*x,y*> = ¬(¬x 'or' ¬y);
  correctness from 2AryBooleDef;
end;

definition
  func xor2 -> Function of 2-tuples_on BOOLEAN, BOOLEAN means:
XOR: for x,y being Element of BOOLEAN holds it.<*x,y*> = x 'xor' y;
  correctness from 2AryBooleDef;
  func xor2a -> Function of 2-tuples_on BOOLEAN, BOOLEAN means:
XOR2A: for x,y being Element of BOOLEAN holds it.<*x,y*> = ¬x 'xor' y;
  correctness from 2AryBooleDef;
  func xor2b -> Function of 2-tuples_on BOOLEAN, BOOLEAN means:
XOR2B: for x,y being Element of BOOLEAN holds it.<*x,y*> = ¬x 'xor' ¬y;
  correctness from 2AryBooleDef;
end;

theorem ::ThAnd2:
  for x,y being Element of BOOLEAN holds and2.<*x,y*> = x '&' y &
   and2a.<*x,y*> = ¬x '&' y & and2b.<*x,y*> = ¬x '&' ¬y
  by AND, AND2A, AND2B;

theorem ::ThNAnd2:
  for x,y being Element of BOOLEAN holds nand2.<*x,y*> = ¬(x '&' y) &
   nand2a.<*x,y*> = ¬(¬x '&' y) & nand2b.<*x,y*> = ¬(¬x '&' ¬y)
  by NAND, NAND2A, NAND2B;

theorem ::ThOr2:
  for x,y being Element of BOOLEAN holds or2.<*x,y*> = x 'or' y &
   or2a.<*x,y*> = ¬x 'or' y & or2b.<*x,y*> = ¬x 'or' ¬y
  by OR, OR2A, OR2B;

theorem ::ThNOr2:
  for x,y being Element of BOOLEAN holds nor2.<*x,y*> = ¬(x 'or' y) &
   nor2a.<*x,y*> = ¬(¬x 'or' y) & nor2b.<*x,y*> = ¬(¬x 'or' ¬y)
  by NOR, NOR2A, NOR2B;
```

```
theorem ::ThXOr2:
  for x,y being Element of BOOLEAN holds xor2.<*x,y*> = x 'xor' y &
   xor2a.<*x,y*> = ¬x 'xor' y & xor2b.<*x,y*> = ¬x 'xor' ¬y
  by XOR, XOR2A, XOR2B;

theorem ::ThAltAnd2:
  for x,y being Element of BOOLEAN holds and2.<*x,y*> = nor2b.<*x,y*> &
   and2a.<*x,y*> = nor2a.<*y,x*> & and2b.<*x,y*> = nor2.<*x,y*>
  proof let x,y be Element of BOOLEAN;
    thus and2.<*x,y*>  = x '&' y          by AND
                       .= ¬(¬x 'or' ¬y)   by BINARITH:12
                       .= nor2b.<*x,y*>   by NOR2B;
    thus and2a.<*x,y*> = ¬x '&' y         by AND2A
                       .= ¬(¬¬x 'or' ¬y)  by BINARITH:12
                       .= ¬(x 'or' ¬y)    by MARGREL1:40
                       .= ¬(¬y 'or' x)    by BINARITH:6
                       .= nor2a.<*y,x*>   by NOR2A;
    thus and2b.<*x,y*> = ¬x '&' ¬y        by AND2B
                       .= ¬(¬¬x 'or' ¬¬y) by BINARITH:12
                       .= ¬(x 'or' ¬¬y)   by MARGREL1:40
                       .= ¬(x 'or' y)     by MARGREL1:40
                       .= nor2.<*x,y*>    by NOR;
  end;

theorem ::ThAltOr2:
  for x,y being Element of BOOLEAN holds or2.<*x,y*> = nand2b.<*x,y*> &
   or2a.<*x,y*> = nand2a.<*y,x*> & or2b.<*x,y*> = nand2.<*x,y*>
  proof let x,y be Element of BOOLEAN;
    thus or2.<*x,y*>  = x 'or' y          by OR
                      .= ¬(¬x '&' ¬y)     by BINARITH:8
                      .= nand2b.<*x,y*>   by NAND2B;
    thus or2a.<*x,y*> = ¬x 'or' y         by OR2A
                      .= ¬(¬¬x '&' ¬y)    by BINARITH:8
                      .= ¬(x '&' ¬y)      by MARGREL1:40
                      .= ¬(¬y '&' x)      by MARGREL1:48
                      .= nand2a.<*y,x*>   by NAND2A;
    thus or2b.<*x,y*> = ¬x 'or' ¬y        by OR2B
                      .= ¬(¬¬x '&' ¬¬y)   by BINARITH:8
                      .= ¬(x '&' ¬¬y)     by MARGREL1:40
                      .= ¬(x '&' y)       by MARGREL1:40
                      .= nand2.<*x,y*>    by NAND;
  end;

theorem ::ThAltXor2:
  for x,y being Element of BOOLEAN holds xor2b.<*x,y*> = xor2.<*x,y*>
  proof let x,y be Element of BOOLEAN;
    thus xor2b.<*x,y*> = ¬x 'xor' ¬y by XOR2B
                       .= (¬¬x '&' ¬y) 'or' (¬x '&' ¬¬y) by BINARITH:def 2
                       .= (x '&' ¬y) 'or' (¬x '&' ¬¬y) by MARGREL1:40
                       .= (x '&' ¬y) 'or' (¬x '&' y) by MARGREL1:40
                       .= (¬x '&' y) 'or' (x '&' ¬y) by BINARITH:6
                       .= x 'xor' y by BINARITH:def 2
                       .= xor2.<*x,y*> by XOR;
  end;

theorem ::ThCalAnd2:
    and2.<*0,0*>=0 &  and2.<*0,1*>=0 &  and2.<*1,0*>=0 &  and2.<*1,1*>=1 &
   and2a.<*0,0*>=0 & and2a.<*0,1*>=1 & and2a.<*1,0*>=0 & and2a.<*1,1*>=0 &
   and2b.<*0,0*>=1 & and2b.<*0,1*>=0 & and2b.<*1,0*>=0 & and2b.<*1,1*>=0
  proof A: TRUE = 1 & FALSE = 0 by MARGREL1:36;
    thus and2.<*0,0*> = FALSE '&' FALSE by A,AND .= 0 by A,MARGREL1:45;
    thus and2.<*0,1*> = FALSE '&' TRUE  by A,AND .= 0 by A,MARGREL1:45;
    thus and2.<*1,0*> = TRUE  '&' FALSE by A,AND .= 0 by A,MARGREL1:45;
    thus and2.<*1,1*> = TRUE  '&' TRUE  by A,AND .= 1 by A,MARGREL1:45;
    thus and2a.<*0,0*> = ¬FALSE '&' FALSE by A,AND2A .= 0 by A,MARGREL1:45;
    thus and2a.<*0,1*> = ¬FALSE '&' TRUE  by A,AND2A
```

```
                    .= TRUE   '&' TRUE  by MARGREL1:41
                    .= 1 by A,MARGREL1:45;
    thus and2a.<*1,0*> = ¬TRUE  '&' FALSE by A,AND2A .= 0 by A,MARGREL1:45;
    thus and2a.<*1,1*> = ¬TRUE  '&' TRUE  by A,AND2A
                    .= FALSE  '&' TRUE  by MARGREL1:41
                    .= 0 by A,MARGREL1:45;
    thus and2b.<*0,0*> = ¬FALSE '&' ¬FALSE by A,AND2B
                    .= TRUE   '&' ¬FALSE by MARGREL1:41
                    .= TRUE   '&' TRUE   by MARGREL1:41
                    .= 1 by A,MARGREL1:45;
    thus and2b.<*0,1*> = ¬FALSE  '&' ¬TRUE by A,AND2B
                    .= ¬FALSE  '&' FALSE by MARGREL1:41
                    .= 0 by A,MARGREL1:45;
    thus and2b.<*1,0*> = ¬TRUE  '&' ¬FALSE by A,AND2B
                    .= FALSE  '&' ¬FALSE by MARGREL1:41
                    .= 0 by A,MARGREL1:45;
    thus and2b.<*1,1*> = ¬TRUE  '&' ¬TRUE  by A,AND2B
                    .= FALSE  '&' ¬TRUE  by MARGREL1:41
                    .= 0 by A,MARGREL1:45;
  end;

theorem ::ThCalOr2:
    or2.<*0,0*>=0 &  or2.<*0,1*>=1 &  or2.<*1,0*>=1 &  or2.<*1,1*>=1 &
   or2a.<*0,0*>=1 & or2a.<*0,1*>=1 & or2a.<*1,0*>=0 & or2a.<*1,1*>=1 &
   or2b.<*0,0*>=1 & or2b.<*0,1*>=1 & or2b.<*1,0*>=1 & or2b.<*1,1*>=0
  proof A: TRUE = 1 & FALSE = 0 by MARGREL1:36;
    thus or2.<*0,0*> = FALSE 'or' FALSE by A,OR .= 0 by A,BINARITH:7;
    thus or2.<*0,1*> = FALSE 'or' TRUE  by A,OR .= 1 by A,BINARITH:19;
    thus or2.<*1,0*> = TRUE  'or' FALSE by A,OR .= 1 by A,BINARITH:19;
    thus or2.<*1,1*> = TRUE  'or' TRUE  by A,OR .= 1 by A,BINARITH:19;
    thus or2a.<*0,0*> = ¬FALSE 'or' FALSE by A,OR2A
                    .= TRUE   'or' FALSE by MARGREL1:41
                    .= 1 by A,BINARITH:19;
    thus or2a.<*0,1*> = ¬FALSE 'or' TRUE  by A,OR2A .= 1 by A,BINARITH:19;
    thus or2a.<*1,0*> = ¬TRUE  'or' FALSE by A,OR2A
                    .= FALSE  'or' FALSE by MARGREL1:41
                    .= 0 by A,BINARITH:7;
    thus or2a.<*1,1*> = ¬TRUE  'or' TRUE  by A,OR2A .= 1 by A,BINARITH:19;
    thus or2b.<*0,0*> = ¬FALSE 'or' ¬FALSE by A,OR2B
                    .= TRUE   'or' ¬FALSE by MARGREL1:41
                    .= 1 by A,BINARITH:19;
    thus or2b.<*0,1*> = ¬FALSE 'or' ¬TRUE  by A,OR2B
                    .= TRUE   'or' ¬TRUE  by MARGREL1:41
                    .= 1 by A,BINARITH:19;
    thus or2b.<*1,0*> = ¬TRUE  'or' ¬FALSE by A,OR2B
                    .= ¬TRUE  'or' TRUE   by MARGREL1:41
                    .= 1 by A,BINARITH:19;
    thus or2b.<*1,1*> = ¬TRUE  'or' ¬TRUE  by A,OR2B
                    .= FALSE  'or' ¬TRUE  by MARGREL1:41
                    .= FALSE  'or' FALSE  by MARGREL1:41
                    .= 0 by A,BINARITH:7;
  end;

theorem ::ThCalXor2:
    xor2.<*0,0*>=0 &  xor2.<*0,1*>=1 &  xor2.<*1,0*>=1 &  xor2.<*1,1*>=0 &
   xor2a.<*0,0*>=1 & xor2a.<*0,1*>=0 & xor2a.<*1,0*>=0 & xor2a.<*1,1*>=1
  proof A: TRUE = 1 & FALSE = 0 by MARGREL1:36;
    thus xor2.<*0,0*> = FALSE 'xor' FALSE by A,XOR .= 0 by A,BINARITH:15;
    thus xor2.<*0,1*> = FALSE 'xor' TRUE  by A,XOR .= 1 by A,BINARITH:14;
    thus xor2.<*1,0*> = TRUE  'xor' FALSE by A,XOR .= 1 by A,BINARITH:14;
    thus xor2.<*1,1*> = TRUE  'xor' TRUE  by A,XOR .= 0 by A,BINARITH:15;
    thus xor2a.<*0,0*> = ¬FALSE 'xor' FALSE by A,XOR2A .= 1 by A,BINARITH:17;
    thus xor2a.<*0,1*> = ¬FALSE 'xor' TRUE  by A,XOR2A
                    .= ¬¬FALSE by BINARITH:13
                    .= 0 by A,MARGREL1:40;
    thus xor2a.<*1,0*> = ¬TRUE  'xor' FALSE by A,XOR2A
```

```
                        .= ¬TRUE by BINARITH:14
                        .= 0 by A,MARGREL1:41;
    thus xor2a.<*1,1*> = ¬TRUE 'xor' TRUE by A,XOR2A
                        .= ¬¬TRUE by BINARITH:13
                        .= 1 by A,MARGREL1:40;
  end;

:: 3-Input Operators

definition
  func and3 -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
AND3: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = x '&' y '&' z; correctness from 3AryBooleDef;
  func and3a -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
AND3A: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = ¬x '&' y '&' z; correctness from 3AryBooleDef;
  func and3b -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
AND3B: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = ¬x '&' ¬y '&' z; correctness from 3AryBooleDef;
  func and3c -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
AND3C: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = ¬x '&' ¬y '&' ¬z; correctness from 3AryBooleDef;
end;

definition
  func nand3 -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
NAND3: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = ¬(x '&' y '&' z); correctness from 3AryBooleDef;
  func nand3a -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
NAND3A: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = ¬(¬x '&' y '&' z); correctness from 3AryBooleDef;
  func nand3b -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
NAND3B: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = ¬(¬x '&' ¬y '&' z); correctness from 3AryBooleDef;
  func nand3c -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
NAND3C: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = ¬(¬x '&' ¬y '&' ¬z); correctness from 3AryBooleDef;
end;

definition
  func or3 -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
OR3: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = x 'or' y 'or' z; correctness from 3AryBooleDef;
  func or3a -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
OR3A: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = ¬x 'or' y 'or' z; correctness from 3AryBooleDef;
  func or3b -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
OR3B: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = ¬x 'or' ¬y 'or' z; correctness from 3AryBooleDef;
  func or3c -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
OR3C: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = ¬x 'or' ¬y 'or' ¬z; correctness from 3AryBooleDef;
end;

definition
  func nor3 -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
NOR3: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = ¬(x 'or' y 'or' z); correctness from 3AryBooleDef;
  func nor3a -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
NOR3A: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = ¬(¬x 'or' y 'or' z); correctness from 3AryBooleDef;
  func nor3b -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
NOR3B: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = ¬(¬x 'or' ¬y 'or' z); correctness from 3AryBooleDef;
  func nor3c -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
NOR3C: for x,y,z being Element of BOOLEAN holds
```

```
   it.<*x,y,z*> = ¬(¬x 'or' ¬y 'or' ¬z); correctness from 3AryBooleDef;
end;

definition
  func xor3 -> Function of 3-tuples_on BOOLEAN, BOOLEAN means:
XOR3: for x,y,z being Element of BOOLEAN holds
   it.<*x,y,z*> = x 'xor' y 'xor' z; correctness from 3AryBooleDef;
end;

theorem ::ThAnd3:
  for x,y,z being Element of BOOLEAN holds and3.<*x,y,z*> = x '&' y '&' z &
   and3a.<*x,y,z*> = ¬x '&' y '&' z & and3b.<*x,y,z*> = ¬x '&' ¬y '&' z &
   and3c.<*x,y,z*> = ¬x '&' ¬y '&' ¬z
  by AND3, AND3A, AND3B, AND3C;

theorem ::ThNAnd3:
  for x,y,z being Element of BOOLEAN holds nand3.<*x,y,z*>=¬(x '&' y '&' z) &
   nand3a.<*x,y,z*>=¬(¬x '&' y '&' z) & nand3b.<*x,y,z*>=¬(¬x '&' ¬y '&' z) &
   nand3c.<*x,y,z*>=¬(¬x '&' ¬y '&' ¬z)
  by NAND3, NAND3A, NAND3B, NAND3C;

theorem ::ThOr3:
  for x,y,z being Element of BOOLEAN holds or3.<*x,y,z*> = x 'or' y 'or' z &
   or3a.<*x,y,z*> = ¬x 'or' y 'or' z & or3b.<*x,y,z*> = ¬x 'or' ¬y 'or' z &
   or3c.<*x,y,z*> = ¬x 'or' ¬y 'or' ¬z
  by OR3, OR3A, OR3B, OR3C;

theorem ::ThNOr3:
  for x,y,z being Element of BOOLEAN holds nor3.<*x,y,z*>=¬(x 'or' y 'or' z) &
   nor3a.<*x,y,z*>=¬(¬x 'or' y 'or' z) & nor3b.<*x,y,z*>=¬(¬x 'or' ¬y 'or' z) &
   nor3c.<*x,y,z*>=¬(¬x 'or' ¬y 'or' ¬z)
  by NOR3, NOR3A, NOR3B, NOR3C;

theorem ::ThXOr3:
  for x,y,z being Element of BOOLEAN holds xor3.<*x,y,z*> = x 'xor' y 'xor' z
  by XOR3;

theorem ::ThAltAnd3:
  for x,y,z being Element of BOOLEAN holds
    and3.<*x,y,z*> = nor3c.<*x,y,z*> & and3a.<*x,y,z*> = nor3b.<*z,y,x*> &
   and3b.<*x,y,z*> = nor3a.<*z,y,x*> & and3c.<*x,y,z*> = nor3.<*x,y,z*>
  proof let x,y,z be Element of BOOLEAN;
    thus and3.<*x,y,z*>  = x '&' y '&' z                 by AND3
                        .= ¬(¬x 'or' ¬y) '&' z           by BINARITH:12
                        .= ¬(¬¬(¬x 'or' ¬y) 'or' ¬z)     by BINARITH:12
                        .= ¬((¬x 'or' ¬y) 'or' ¬z)       by MARGREL1:40
                        .= nor3c.<*x,y,z*>               by NOR3C;
    thus and3a.<*x,y,z*> = ¬x '&' y '&' z                by AND3A
                        .= ¬(¬¬x 'or' ¬y) '&' z          by BINARITH:12
                        .= ¬(¬¬(¬¬x 'or' ¬y) 'or' ¬z)    by BINARITH:12
                        .= ¬((¬¬x 'or' ¬y) 'or' ¬z)      by MARGREL1:40
                        .= ¬((x 'or' ¬y) 'or' ¬z)        by MARGREL1:40
                        .= ¬((¬y 'or' x) 'or' ¬z)        by BINARITH:6
                        .= ¬(¬z 'or' (¬y 'or' x))        by BINARITH:6
                        .= ¬((¬z 'or' ¬y) 'or' x)        by BINARITH:20
                        .= nor3b.<*z,y,x*>               by NOR3B;
    thus and3b.<*x,y,z*> = ¬x '&' ¬y '&' z               by AND3B
                        .= ¬(¬¬x 'or' ¬¬y) '&' z         by BINARITH:12
                        .= ¬(¬¬(¬¬x 'or' ¬¬y) 'or' ¬z)   by BINARITH:12
                        .= ¬((¬¬x 'or' ¬¬y) 'or' ¬z)     by MARGREL1:40
                        .= ¬((x 'or' ¬¬y) 'or' ¬z)       by MARGREL1:40
                        .= ¬((x 'or' y) 'or' ¬z)         by MARGREL1:40
                        .= ¬((y 'or' x) 'or' ¬z)         by BINARITH:6
                        .= ¬(¬z 'or' (y 'or' x))         by BINARITH:6
                        .= ¬((¬z 'or' y) 'or' x)         by BINARITH:20
                        .= nor3a.<*z,y,x*>               by NOR3A;
```

```
    thus and3c.<*x,y,z*> = ¬x '&' ¬y '&' ¬z                    by AND3C
                        .= ¬(¬¬x 'or' ¬¬y) '&' ¬z              by BINARITH:12
                        .= ¬(¬¬(¬¬x 'or' ¬¬y) 'or' ¬¬z)        by BINARITH:12
                        .= ¬((¬¬x 'or' ¬¬y) 'or' ¬¬z)          by MARGREL1:40
                        .= ¬((x 'or' ¬¬y) 'or' ¬¬z)            by MARGREL1:40
                        .= ¬((x 'or' y) 'or' ¬¬z)              by MARGREL1:40
                        .= ¬((x 'or' y) 'or' z)                by MARGREL1:40
                        .= nor3.<*x,y,z*>                      by NOR3;
  end;

theorem ::ThAltOr3:
  for x,y,z being Element of BOOLEAN holds
    or3.<*x,y,z*> = nand3c.<*x,y,z*> & or3a.<*x,y,z*> = nand3b.<*z,y,x*> &
   or3b.<*x,y,z*> = nand3a.<*z,y,x*> & or3c.<*x,y,z*> = nand3.<*x,y,z*>
  proof let x,y,z be Element of BOOLEAN;
    thus or3.<*x,y,z*>  = x 'or' y 'or' z                      by OR3
                        .= ¬(¬x '&' ¬y) 'or' z                 by BINARITH:8
                        .= ¬(¬¬(¬x '&' ¬y) '&' ¬z)             by BINARITH:8
                        .= ¬((¬x '&' ¬y) '&' ¬z)               by MARGREL1:40
                        .= nand3c.<*x,y,z*>                    by NAND3C;
    thus or3a.<*x,y,z*> = ¬x 'or' y 'or' z                     by OR3A
                        .= ¬(¬¬x '&' ¬y) 'or' z                by BINARITH:8
                        .= ¬(¬¬(¬¬x '&' ¬y) '&' ¬z)            by BINARITH:8
                        .= ¬((¬¬x '&' ¬y) '&' ¬z)              by MARGREL1:40
                        .= ¬((x '&' ¬y) '&' ¬z)                by MARGREL1:40
                        .= ¬(¬z '&' ¬y '&' x)                  by BINARITH:31
                        .= nand3b.<*z,y,x*>                    by NAND3B;
    thus or3b.<*x,y,z*> = ¬x 'or' ¬y 'or' z                    by OR3B
                        .= ¬(¬¬x '&' ¬¬y) 'or' z               by BINARITH:8
                        .= ¬(¬¬(¬¬x '&' ¬¬y) '&' ¬z)           by BINARITH:8
                        .= ¬((¬¬x '&' ¬¬y) '&' ¬z)             by MARGREL1:40
                        .= ¬((x '&' ¬¬y) '&' ¬z)               by MARGREL1:40
                        .= ¬((x '&' y) '&' ¬z)                 by MARGREL1:40
                        .= ¬(¬z '&' y '&' x)                   by BINARITH:31
                        .= nand3a.<*z,y,x*>                    by NAND3A;
    thus or3c.<*x,y,z*> = ¬x 'or' ¬y 'or' ¬z                   by OR3C
                        .= ¬(¬¬x '&' ¬¬y) 'or' ¬z              by BINARITH:8
                        .= ¬(¬¬(¬¬x '&' ¬¬y) '&' ¬¬z)          by BINARITH:8
                        .= ¬((¬¬x '&' ¬¬y) '&' ¬¬z)            by MARGREL1:40
                        .= ¬((x '&' ¬¬y) '&' ¬¬z)              by MARGREL1:40
                        .= ¬((x '&' y) '&' ¬¬z)                by MARGREL1:40
                        .= ¬((x '&' y) '&' z)                  by MARGREL1:40
                        .= nand3.<*x,y,z*>                     by NAND3;
  end;

theorem ::ThCalAnd3:
    and3.<*0,0,0*>=0 & and3.<*0,0,1*>=0 & and3.<*0,1,0*>=0 &
    and3.<*0,1,1*>=0 & and3.<*1,0,0*>=0 & and3.<*1,0,1*>=0 &
    and3.<*1,1,0*>=0 & and3.<*1,1,1*>=1
  proof A: TRUE = 1 & FALSE = 0 by MARGREL1:36;
    thus and3.<*0,0,0*> = FALSE '&' FALSE '&' FALSE by A,AND3
                        .= 0 by A,MARGREL1:45;
    thus and3.<*0,0,1*> = FALSE '&' FALSE '&' TRUE by A,AND3
                        .= FALSE '&' TRUE by MARGREL1:45
                        .= 0 by A,MARGREL1:45;
    thus and3.<*0,1,0*> = FALSE '&' TRUE '&' FALSE by A,AND3
                        .= 0 by A,MARGREL1:45;
    thus and3.<*0,1,1*> = FALSE '&' TRUE '&' TRUE by A,AND3
                        .= FALSE '&' TRUE by MARGREL1:45
                        .= 0 by A,MARGREL1:45;
    thus and3.<*1,0,0*> = TRUE  '&' FALSE '&' FALSE by A,AND3
                        .= 0 by A,MARGREL1:45;
    thus and3.<*1,0,1*> = TRUE  '&' FALSE '&' TRUE by A,AND3
                        .= FALSE '&' TRUE by MARGREL1:45
                        .= 0 by A,MARGREL1:45;
    thus and3.<*1,1,0*> = TRUE  '&' TRUE '&' FALSE by A,AND3
```

```
                        .= 0 by A,MARGREL1:45;
    thus and3.<*1,1,1*> = TRUE  '&' TRUE '&' TRUE by A,AND3
                        .= TRUE  '&' TRUE by MARGREL1:45
                        .= 1 by A,MARGREL1:45;
  end;

theorem ::ThCalAnd3_a:
    and3a.<*0,0,0*>=0 & and3a.<*0,0,1*>=0 & and3a.<*0,1,0*>=0 &
    and3a.<*0,1,1*>=1 & and3a.<*1,0,0*>=0 & and3a.<*1,0,1*>=0 &
    and3a.<*1,1,0*>=0 & and3a.<*1,1,1*>=0
  proof A: TRUE = 1 & FALSE = 0 by MARGREL1:36;
    thus and3a.<*0,0,0*> = ¬FALSE '&' FALSE '&' FALSE by A,AND3A
                        .= 0 by A,MARGREL1:45;
    thus and3a.<*0,0,1*> = ¬FALSE '&' FALSE '&' TRUE by A,AND3A
                        .= FALSE '&' TRUE by MARGREL1:45
                        .= 0 by A,MARGREL1:45;
    thus and3a.<*0,1,0*> = ¬FALSE '&' TRUE '&' FALSE by A,AND3A
                        .= 0 by A,MARGREL1:45;
    thus and3a.<*0,1,1*> = ¬FALSE '&' TRUE '&' TRUE by A,AND3A
                        .= (TRUE '&' ¬FALSE) '&' TRUE by MARGREL1:48
                        .= ¬FALSE '&' TRUE by MARGREL1:50
                        .= TRUE  '&' TRUE by MARGREL1:41
                        .= 1 by A,MARGREL1:45;
    thus and3a.<*1,0,0*> = ¬TRUE  '&' FALSE '&' FALSE by A,AND3A
                        .= 0 by A,MARGREL1:45;
    thus and3a.<*1,0,1*> = ¬TRUE  '&' FALSE '&' TRUE by A,AND3A
                        .= FALSE '&' TRUE by MARGREL1:45
                        .= 0 by A,MARGREL1:45;
    thus and3a.<*1,1,0*> = ¬TRUE  '&' TRUE '&' FALSE by A,AND3A
                        .= 0 by A,MARGREL1:45;
    thus and3a.<*1,1,1*> = ¬TRUE  '&' TRUE '&' TRUE by A,AND3A
                        .= (TRUE '&' ¬TRUE) '&' TRUE by MARGREL1:48
                        .= ¬TRUE '&' TRUE by MARGREL1:50
                        .= FALSE '&' TRUE by MARGREL1:41
                        .= 0 by A,MARGREL1:45;
  end;

theorem ::ThCalAnd3_b:
    and3b.<*0,0,0*>=0 & and3b.<*0,0,1*>=1 & and3b.<*0,1,0*>=0 &
    and3b.<*0,1,1*>=0 & and3b.<*1,0,0*>=0 & and3b.<*1,0,1*>=0 &
    and3b.<*1,1,0*>=0 & and3b.<*1,1,1*>=0
  proof A: TRUE = 1 & FALSE = 0 by MARGREL1:36;
    thus and3b.<*0,0,0*> = ¬FALSE '&' ¬FALSE '&' FALSE by A,AND3B
                        .= 0 by A,MARGREL1:45;
    thus and3b.<*0,0,1*> = ¬FALSE '&' ¬FALSE '&' TRUE by A,AND3B
                        .= TRUE '&' ¬FALSE '&' TRUE by MARGREL1:41
                        .= TRUE '&' TRUE '&' TRUE by MARGREL1:41
                        .= TRUE '&' TRUE by MARGREL1:45
                        .= 1 by A,MARGREL1:45;
    thus and3b.<*0,1,0*> = ¬FALSE '&' ¬TRUE '&' FALSE by A,AND3B
                        .= 0 by A,MARGREL1:45;
    thus and3b.<*0,1,1*> = ¬FALSE '&' ¬TRUE '&' TRUE by A,AND3B
                        .= ¬FALSE '&' FALSE '&' TRUE by MARGREL1:41
                        .= FALSE '&' TRUE by MARGREL1:45
                        .= 0 by A,MARGREL1:45;
    thus and3b.<*1,0,0*> = ¬TRUE  '&' ¬FALSE '&' FALSE by A,AND3B
                        .= 0 by A,MARGREL1:45;
    thus and3b.<*1,0,1*> = ¬TRUE  '&' ¬FALSE '&' TRUE by A,AND3B
                        .= FALSE '&' ¬FALSE '&' TRUE by MARGREL1:41
                        .= FALSE '&' TRUE by MARGREL1:45
                        .= 0 by A,MARGREL1:45;
    thus and3b.<*1,1,0*> = ¬TRUE  '&' ¬TRUE '&' FALSE by A,AND3B
                        .= 0 by A,MARGREL1:45;
    thus and3b.<*1,1,1*> = ¬TRUE  '&' ¬TRUE '&' TRUE by A,AND3B
                        .= FALSE '&' ¬TRUE '&' TRUE by MARGREL1:41
                        .= FALSE '&' TRUE by MARGREL1:45
```

```
                        .= 0 by A,MARGREL1:45;
  end;

theorem ::ThCalAnd3_c:
    and3c.<*0,0,0*>=1 & and3c.<*0,0,1*>=0 & and3c.<*0,1,0*>=0 &
    and3c.<*0,1,1*>=0 & and3c.<*1,0,0*>=0 & and3c.<*1,0,1*>=0 &
    and3c.<*1,1,0*>=0 & and3c.<*1,1,1*>=0
  proof A: TRUE = 1 & FALSE = 0 by MARGREL1:36;
    thus and3c.<*0,0,0*> = ¬FALSE '&' ¬FALSE '&' ¬FALSE by A,AND3C
                        .= TRUE '&' ¬FALSE '&' ¬FALSE by MARGREL1:41
                        .= TRUE '&' TRUE '&' ¬FALSE by MARGREL1:41
                        .= TRUE '&' TRUE '&' TRUE by MARGREL1:41
                        .= TRUE '&' TRUE by MARGREL1:45
                        .= 1 by A,MARGREL1:45;
    thus and3c.<*0,0,1*> = ¬FALSE '&' ¬FALSE '&' ¬TRUE by A,AND3C
                        .= ¬FALSE '&' ¬FALSE '&' FALSE by MARGREL1:41
                        .= 0 by A,MARGREL1:45;
    thus and3c.<*0,1,0*> = ¬FALSE '&' ¬TRUE '&' ¬FALSE by A,AND3C
                        .= ¬FALSE '&' FALSE '&' ¬FALSE by MARGREL1:41
                        .= ¬FALSE '&' ¬FALSE '&' FALSE by BINARITH:32
                        .= 0 by A,MARGREL1:45;
    thus and3c.<*0,1,1*> = ¬FALSE '&' ¬TRUE '&' ¬TRUE by A,AND3C
                        .= ¬FALSE '&' ¬TRUE '&' FALSE by MARGREL1:41
                        .= 0 by A,MARGREL1:45;
    thus and3c.<*1,0,0*> = ¬TRUE  '&' ¬FALSE '&' ¬FALSE by A,AND3C
                        .= FALSE  '&' ¬FALSE '&' ¬FALSE by MARGREL1:41
                        .= ¬FALSE '&' ¬FALSE '&'  FALSE by BINARITH:30
                        .= 0 by A,MARGREL1:45;
    thus and3c.<*1,0,1*> = ¬TRUE  '&' ¬FALSE '&' ¬TRUE by A,AND3C
                        .= ¬TRUE  '&' ¬FALSE '&' FALSE by MARGREL1:41
                        .= 0 by A,MARGREL1:45;
    thus and3c.<*1,1,0*> = ¬TRUE  '&' ¬TRUE '&' ¬FALSE by A,AND3C
                        .= FALSE  '&' ¬TRUE '&' ¬FALSE by MARGREL1:41
                        .= ¬TRUE  '&' ¬FALSE '&' FALSE by BINARITH:30
                        .= 0 by A,MARGREL1:45;
    thus and3c.<*1,1,1*> = ¬TRUE  '&' ¬TRUE '&' ¬TRUE by A,AND3C
                        .= FALSE  '&' ¬TRUE '&' ¬TRUE by MARGREL1:41
                        .= ¬TRUE  '&' ¬TRUE '&' FALSE by BINARITH:30
                        .= 0 by A,MARGREL1:45;
  end;

theorem ::ThCalOr3:
    or3.<*0,0,0*> = 0 & or3.<*0,0,1*> = 1 & or3.<*0,1,0*> = 1 &
    or3.<*0,1,1*> = 1 & or3.<*1,0,0*> = 1 & or3.<*1,0,1*> = 1 &
    or3.<*1,1,0*> = 1 & or3.<*1,1,1*> = 1
  proof A: TRUE = 1 & FALSE = 0 by MARGREL1:36;
    thus or3.<*0,0,0*>   = FALSE 'or' FALSE 'or' FALSE by A,OR3
                        .= FALSE 'or' FALSE by BINARITH:7
                        .= 0 by A,BINARITH:7;
    thus or3.<*0,0,1*>   = FALSE 'or' FALSE 'or' TRUE by A,OR3
                        .= 1 by A,BINARITH:19;
    thus or3.<*0,1,0*>   = FALSE 'or' TRUE 'or' FALSE by A,OR3
                        .= TRUE 'or' FALSE by BINARITH:19
                        .= 1 by A,BINARITH:19;
    thus or3.<*0,1,1*>   = FALSE 'or' TRUE 'or' TRUE by A,OR3
                        .= 1 by A,BINARITH:19;
    thus or3.<*1,0,0*>   = TRUE 'or' FALSE 'or' FALSE by A,OR3
                        .= TRUE 'or' FALSE by BINARITH:7
                        .= 1 by A,BINARITH:19;
    thus or3.<*1,0,1*>   = TRUE 'or' FALSE 'or' TRUE by A,OR3
                        .= 1 by A,BINARITH:19;
    thus or3.<*1,1,0*>   = TRUE 'or' TRUE 'or' FALSE by A,OR3
                        .= TRUE 'or' FALSE by BINARITH:19
                        .= 1 by A,BINARITH:19;
    thus or3.<*1,1,1*>   = TRUE 'or' TRUE 'or' TRUE by A,OR3
                        .= 1 by A,BINARITH:19;
```

```
  end;

theorem ::ThCalOr3_a:
    or3a.<*0,0,0*> = 1 & or3a.<*0,0,1*> = 1 & or3a.<*0,1,0*> = 1 &
    or3a.<*0,1,1*> = 1 & or3a.<*1,0,0*> = 0 & or3a.<*1,0,1*> = 1 &
    or3a.<*1,1,0*> = 1 & or3a.<*1,1,1*> = 1
  proof A: TRUE = 1 & FALSE = 0 by MARGREL1:36;
    thus or3a.<*0,0,0*>  = ¬FALSE 'or' FALSE 'or' FALSE by A,OR3A
                        .= ¬FALSE 'or' FALSE by BINARITH:7
                        .= TRUE 'or' FALSE by MARGREL1:41
                        .= 1 by A,BINARITH:19;
    thus or3a.<*0,0,1*>  = ¬FALSE 'or' FALSE 'or' TRUE by A,OR3A
                        .= 1 by A,BINARITH:19;
    thus or3a.<*0,1,0*>  = ¬FALSE 'or' TRUE 'or' FALSE by A,OR3A
                        .= TRUE 'or' FALSE by BINARITH:19
                        .= 1 by A,BINARITH:19;
    thus or3a.<*0,1,1*>  = ¬FALSE 'or' TRUE 'or' TRUE by A,OR3A
                        .= 1 by A,BINARITH:19;
    thus or3a.<*1,0,0*>  = ¬TRUE 'or' FALSE 'or' FALSE by A,OR3A
                        .= ¬TRUE 'or' FALSE by BINARITH:7
                        .= FALSE 'or' FALSE by MARGREL1:41
                        .= 0 by A,BINARITH:7;
    thus or3a.<*1,0,1*>  = ¬TRUE 'or' FALSE 'or' TRUE by A,OR3A
                        .= 1 by A,BINARITH:19;
    thus or3a.<*1,1,0*>  = ¬TRUE 'or' TRUE 'or' FALSE by A,OR3A
                        .= TRUE 'or' FALSE by BINARITH:19
                        .= 1 by A,BINARITH:19;
    thus or3a.<*1,1,1*>  = ¬TRUE 'or' TRUE 'or' TRUE by A,OR3A
                        .= 1 by A,BINARITH:19;
  end;

theorem ::ThCalOr3_b:
    or3b.<*0,0,0*> = 1 & or3b.<*0,0,1*> = 1 & or3b.<*0,1,0*> = 1 &
    or3b.<*0,1,1*> = 1 & or3b.<*1,0,0*> = 1 & or3b.<*1,0,1*> = 1 &
    or3b.<*1,1,0*> = 0 & or3b.<*1,1,1*> = 1
  proof A: TRUE = 1 & FALSE = 0 by MARGREL1:36;
    thus or3b.<*0,0,0*>  = ¬FALSE 'or' ¬FALSE 'or' FALSE by A,OR3B
                        .= TRUE 'or' ¬FALSE 'or' FALSE by MARGREL1:41
                        .= TRUE 'or' FALSE by BINARITH:19
                        .= 1 by A,BINARITH:19;
    thus or3b.<*0,0,1*>  = ¬FALSE 'or' ¬FALSE 'or' TRUE by A,OR3B
                        .= 1 by A,BINARITH:19;
    thus or3b.<*0,1,0*>  = ¬FALSE 'or' ¬TRUE 'or' FALSE by A,OR3B
                        .= ¬FALSE 'or' ¬TRUE by BINARITH:7
                        .= TRUE 'or' ¬TRUE by MARGREL1:41
                        .= 1 by A,BINARITH:19;
    thus or3b.<*0,1,1*>  = ¬FALSE 'or' ¬TRUE 'or' TRUE by A,OR3B
                        .= 1 by A,BINARITH:19;
    thus or3b.<*1,0,0*>  = ¬TRUE 'or' ¬FALSE 'or' FALSE by A,OR3B
                        .= ¬TRUE 'or' ¬FALSE by BINARITH:7
                        .= ¬TRUE 'or' TRUE by MARGREL1:41
                        .= 1 by A,BINARITH:19;
    thus or3b.<*1,0,1*>  = ¬TRUE 'or' ¬FALSE 'or' TRUE by A,OR3B
                        .= 1 by A,BINARITH:19;
    thus or3b.<*1,1,0*>  = ¬TRUE 'or' ¬TRUE 'or' FALSE by A,OR3B
                        .= ¬TRUE 'or' ¬TRUE by BINARITH:7
                        .= FALSE 'or' ¬TRUE by MARGREL1:41
                        .= FALSE 'or' FALSE by MARGREL1:41
                        .= 0 by A,BINARITH:7;
    thus or3b.<*1,1,1*>  = ¬TRUE 'or' ¬TRUE 'or' TRUE by A,OR3B
                        .= 1 by A,BINARITH:19;
  end;

theorem ::ThCalOr3_c:
    or3c.<*0,0,0*> = 1 & or3c.<*0,0,1*> = 1 & or3c.<*0,1,0*> = 1 &
```

```
    or3c.<*0,1,1*> = 1 & or3c.<*1,0,0*> = 1 & or3c.<*1,0,1*> = 1 &
    or3c.<*1,1,0*> = 1 & or3c.<*1,1,1*> = 0
  proof A: TRUE = 1 & FALSE = 0 by MARGREL1:36;
    thus or3c.<*0,0,0*>  = ¬FALSE 'or' ¬FALSE 'or' ¬FALSE by A,OR3C
                        .= ¬FALSE 'or' ¬FALSE 'or' TRUE by MARGREL1:41
                        .= 1 by A,BINARITH:19;
    thus or3c.<*0,0,1*>  = ¬FALSE 'or' ¬FALSE 'or' ¬TRUE by A,OR3C
                        .= ¬FALSE 'or' ¬FALSE 'or' FALSE by MARGREL1:41
                        .= ¬FALSE 'or' ¬FALSE by BINARITH:7
                        .= TRUE 'or' ¬FALSE by MARGREL1:41
                        .= 1 by A,BINARITH:19;
    thus or3c.<*0,1,0*>  = ¬FALSE 'or' ¬TRUE 'or' ¬FALSE by A,OR3C
                        .= ¬FALSE 'or' ¬TRUE 'or' TRUE by MARGREL1:41
                        .= 1 by A,BINARITH:19;
    thus or3c.<*0,1,1*>  = ¬FALSE 'or' ¬TRUE 'or' ¬TRUE by A,OR3C
                        .= TRUE 'or' ¬TRUE 'or' ¬TRUE by MARGREL1:41
                        .= TRUE 'or' ¬TRUE by BINARITH:19
                        .= 1 by A,BINARITH:19;
    thus or3c.<*1,0,0*>  = ¬TRUE 'or' ¬FALSE 'or' ¬FALSE by A,OR3C
                        .= ¬TRUE 'or' ¬FALSE 'or' TRUE by MARGREL1:41
                        .= 1 by A,BINARITH:19;
    thus or3c.<*1,0,1*>  = ¬TRUE 'or' ¬FALSE 'or' ¬TRUE by A,OR3C
                        .= ¬TRUE 'or' TRUE 'or' ¬TRUE by MARGREL1:41
                        .= TRUE 'or' ¬TRUE by BINARITH:19
                        .= 1 by A,BINARITH:19;
    thus or3c.<*1,1,0*>  = ¬TRUE 'or' ¬TRUE 'or' ¬FALSE by A,OR3C
                        .= ¬TRUE 'or' ¬TRUE 'or' TRUE by MARGREL1:41
                        .= 1 by A,BINARITH:19;
    thus or3c.<*1,1,1*>  = ¬TRUE 'or' ¬TRUE 'or' ¬TRUE by A,OR3C
                        .= FALSE 'or' ¬TRUE 'or' ¬TRUE by MARGREL1:41
                        .= FALSE 'or' FALSE 'or' ¬TRUE by MARGREL1:41
                        .= FALSE 'or' FALSE 'or' FALSE by MARGREL1:41
                        .= FALSE 'or' FALSE by BINARITH:7
                        .= 0 by A,BINARITH:7;
  end;

theorem ::ThCalXOr3:
    xor3.<*0,0,0*> = 0 & xor3.<*0,0,1*> = 1 & xor3.<*0,1,0*> = 1 &
    xor3.<*0,1,1*> = 0 & xor3.<*1,0,0*> = 1 & xor3.<*1,0,1*> = 0 &
    xor3.<*1,1,0*> = 0 & xor3.<*1,1,1*> = 1
  proof A: TRUE = 1 & FALSE = 0 by MARGREL1:36;
    thus xor3.<*0,0,0*>  = FALSE 'xor' FALSE 'xor' FALSE by A,XOR3
                        .= FALSE 'xor' FALSE by BINARITH:15
                        .= 0 by A,BINARITH:15;
    thus xor3.<*0,0,1*>  = FALSE 'xor' FALSE 'xor' TRUE by A,XOR3
                        .= FALSE 'xor' TRUE by BINARITH:15
                        .= 1 by A,BINARITH:33;
    thus xor3.<*0,1,0*>  = FALSE 'xor' TRUE 'xor' FALSE by A,XOR3
                        .= TRUE 'xor' FALSE by BINARITH:33
                        .= 1 by A,BINARITH:33;
    thus xor3.<*0,1,1*>  = FALSE 'xor' TRUE 'xor' TRUE by A,XOR3
                        .= TRUE 'xor' TRUE by BINARITH:33
                        .= 0 by A,BINARITH:15;
    thus xor3.<*1,0,0*>  = TRUE 'xor' FALSE 'xor' FALSE by A,XOR3
                        .= TRUE 'xor' FALSE by BINARITH:33
                        .= 1 by A,BINARITH:33;
    thus xor3.<*1,0,1*>  = TRUE 'xor' FALSE 'xor' TRUE by A,XOR3
                        .= TRUE 'xor' TRUE by BINARITH:33
                        .= 0 by A,BINARITH:15;
    thus xor3.<*1,1,0*>  = TRUE 'xor' TRUE 'xor' FALSE by A,XOR3
                        .= FALSE 'xor' FALSE by BINARITH:15
                        .= 0 by A,BINARITH:15;
    thus xor3.<*1,1,1*>  = TRUE 'xor' TRUE 'xor' TRUE by A,XOR3
                        .= FALSE 'xor' TRUE by BINARITH:15
                        .= 1 by A,BINARITH:33;
  end;
```

```
::-------------------------------------------------------------------------
:: 1bit 2's Complement Circuit (Complementor + Incrementor)
::-------------------------------------------------------------------------

:: Complementor

definition
 let x,b be set;
 func CompStr(x,b) -> unsplit gate'1=arity gate'2isBoolean
   non void strict non empty ManySortedSign means:
COMPSTR:
  it = 1GateCircStr(<*x,b*>,xor2a);
  correctness;
end;

definition
 let x,b be set;
A: CompStr(x,b) = 1GateCircStr(<*x,b*>,xor2a) by COMPSTR;
 func CompCirc(x,b) ->
   strict Boolean gate'2=den Circuit of CompStr(x,b) means
::COMPCIRC:
  it = 1GateCircuit(x,b,xor2a);
  uniqueness;
  existence by A;
end;

definition
 let x,b be set;
A: CompStr(x,b) = 1GateCircStr(<*x,b*>,xor2a) by COMPSTR;
 func CompOutput(x,b) -> Element of InnerVertices CompStr(x,b) means:
COMPOUT:
  it = [<*x,b*>,xor2a];
  uniqueness;
  existence
   proof
     set p = <*x,b*>;
     [p,xor2a] ∈ InnerVertices 1GateCircStr(p,xor2a) by FACIRC_1:47; then
     [p,xor2a] ∈ InnerVertices CompStr(x,b) by A;
    hence thesis;
   end;
end;

:: Incrementor

definition
 let x,b be set;
 func IncrementStr(x,b) -> unsplit gate'1=arity gate'2isBoolean
   non void strict non empty ManySortedSign means:
INCSTR:
  it = 1GateCircStr(<*x,b*>,and2a);
  correctness;
end;

definition
 let x,b be set;
A: IncrementStr(x,b) = 1GateCircStr(<*x,b*>,and2a) by INCSTR;
 func IncrementCirc(x,b) ->
   strict Boolean gate'2=den Circuit of IncrementStr(x,b) means
::INCCIRC:
  it = 1GateCircuit(x,b,and2a);
  uniqueness;
  existence by A;
end;

definition
```

```
 let x,b be set;
A: IncrementStr(x,b) = 1GateCircStr(<*x,b*>,and2a) by INCSTR;
 func IncrementOutput(x,b) -> Element of InnerVertices IncrementStr(x,b) means:
INCOUT:
  it = [<*x,b*>,and2a];
  uniqueness;
  existence
   proof
     set p = <*x,b*>;
     [p,and2a] ∈ InnerVertices 1GateCircStr(p,and2a) by FACIRC_1:47; then
     [p,and2a] ∈ InnerVertices IncrementStr(x,b) by A;
    hence thesis;
   end;
end;

:: 2's-BitComplementor

definition
 let x,b be set;
 func BitCompStr(x,b) -> unsplit gate'1=arity gate'2isBoolean
   non void strict non empty ManySortedSign means:
BITCOMPSTR:
  it =  CompStr(x,b) +· IncrementStr(x,b);
  correctness;
end;

definition
 let x,b be set;
A: BitCompStr(x,b) = CompStr(x,b) +· IncrementStr(x,b) by BITCOMPSTR;
 func BitCompCirc(x,b) ->
   strict Boolean gate'2=den Circuit of BitCompStr(x,b) means
  it =  CompCirc(x,b) +· IncrementCirc(x,b);
  uniqueness;
  existence by A;
end;

:: Relation, carrier, InnerVertices, InputVertices and without_pair

:: Complementor

theorem ThCOMPIV:
 for x,b being non pair set holds InnerVertices CompStr(x,b) is Relation
   proof
   let x,b be non pair set;
     CompStr(x,b) = 1GateCircStr(<*x,b*>,xor2a) by COMPSTR; then
     InnerVertices CompStr(x,b) is Relation by FACIRC_1:38;
   hence thesis;
  end;

theorem ThCOMPCA:
 for x,b being non pair set holds
   x ∈ the carrier of CompStr(x,b) &
   b ∈ the carrier of CompStr(x,b) &
   [<*x,b*>,xor2a] ∈ the carrier of CompStr(x,b)
  proof
   let x,b be non pair set;
   set S = CompStr(x,b);
    S = 1GateCircStr(<*x,b*>,xor2a) by COMPSTR; then
    x ∈ the carrier of S & b ∈ the carrier of S &
    [<*x,b*>,xor2a] ∈ the carrier of S by FACIRC_1:43;
   hence thesis;
 end;

theorem ThCOMPCA':
 for x,b being non pair set holds
  the carrier of CompStr(x,b) = {x,b} U {[<*x,b*>,xor2a]}
```

```
  proof
   let x,b be non pair set;
   set p = <*x,b*>;
   set S = CompStr(x,b);
A:  rng p = {x,b} by FINSEQ_3:35;
    the carrier of S = the carrier of 1GateCircStr(p,xor2a) by COMPSTR
                     .= {x,b} U {[p,xor2a]} by A,CIRCCOMB:def 6;
   hence thesis;
 end;

theorem ThCOMPF1:
 for x,b being non pair set holds
   InnerVertices CompStr(x,b) = {[<*x,b*>,xor2a]}
   proof
    let x,b be non pair set;
    set p = <*x,b*>;
    set S = CompStr(x,b);
     InnerVertices S = InnerVertices 1GateCircStr(p,xor2a) by COMPSTR
                     .= {[p,xor2a]} by CIRCCOMB:49;
   hence thesis;
 end;

theorem ThCOMPF1':
 for x,b being non pair set holds
   [<*x,b*>,xor2a] ∈ InnerVertices CompStr(x,b)
  proof
   let x,b be non pair set;
    InnerVertices CompStr(x,b) = {[<*x,b*>,xor2a]} by ThCOMPF1;
   hence thesis by TARSKI:def 1;
 end;

theorem ThCOMPF2:
 for x,b being non pair set holds
  InputVertices CompStr(x,b) = {x,b}
  proof
    let x,b be non pair set;
    set p = <*x,b*>;
    set S = CompStr(x,b);
     S = 1GateCircStr(p,xor2a) by COMPSTR; then
     InputVertices S = {x,b} by FACIRC_1:40;
   hence thesis;
 end;

theorem ::ThCOMPF2':
 for x,b being non pair set holds
  x ∈ InputVertices CompStr(x,b) &
  b ∈ InputVertices CompStr(x,b)
  proof
   let x,b be non pair set;
    InputVertices CompStr(x,b) = {x,b} by ThCOMPF2;
   hence thesis by ENUMSET1:9;
  end;

theorem ThCOMPW:
 for x,b being non pair set holds
  InputVertices CompStr(x,b) is without_pairs
  proof
   let x,b be non pair set;
    InputVertices CompStr(x,b) = {x,b} by ThCOMPF2;
   hence thesis;
  end;

:: Incrementor

theorem ThINCIV:
 for x,b being non pair set holds InnerVertices IncrementStr(x,b) is Relation
```

```
   proof
   let x,b be non pair set;
    IncrementStr(x,b) = 1GateCircStr(<*x,b*>,and2a) by INCSTR; then
     InnerVertices IncrementStr(x,b) is Relation by FACIRC_1:38;
   hence thesis;
  end;

theorem ThINCCA:
 for x,b being non pair set holds
   x ∈ the carrier of IncrementStr(x,b) &
   b ∈ the carrier of IncrementStr(x,b) &
   [<*x,b*>,and2a] ∈ the carrier of IncrementStr(x,b)
  proof
   let x,b be non pair set;
   set S = IncrementStr(x,b);
    S = 1GateCircStr(<*x,b*>,and2a) by INCSTR; then
    x ∈ the carrier of S & b ∈ the carrier of S &
    [<*x,b*>,and2a] ∈ the carrier of S by FACIRC_1:43;
   hence thesis;
 end;

theorem ThINCCA':
 for x,b being non pair set holds
  the carrier of IncrementStr(x,b) = {x,b} U {[<*x,b*>,and2a]}
  proof
   let x,b be non pair set;
   set p = <*x,b*>;
   set S = IncrementStr(x,b);
A:  rng p = {x,b} by FINSEQ_3:35;
    the carrier of S = the carrier of 1GateCircStr(p,and2a) by INCSTR
                    .= {x,b} U {[p,and2a]} by A,CIRCCOMB:def 6;
   hence thesis;
 end;

theorem ThINCF1:
 for x,b being non pair set holds
   InnerVertices IncrementStr(x,b) = {[<*x,b*>,and2a]}
   proof
    let x,b be non pair set;
    set p = <*x,b*>;
    set S = IncrementStr(x,b);
     InnerVertices S = InnerVertices 1GateCircStr(p,and2a) by INCSTR
                    .= {[p,and2a]} by CIRCCOMB:49;
   hence thesis;
 end;

theorem ThINCF1':
 for x,b being non pair set holds
   [<*x,b*>,and2a] ∈ InnerVertices IncrementStr(x,b)
  proof
   let x,b be non pair set;
    InnerVertices IncrementStr(x,b) = {[<*x,b*>,and2a]} by ThINCF1;
   hence thesis by TARSKI:def 1;
 end;

theorem ThINCF2:
 for x,b being non pair set holds
  InputVertices IncrementStr(x,b) = {x,b}
  proof
    let x,b be non pair set;
    set p = <*x,b*>;
    set S = IncrementStr(x,b);
     S = 1GateCircStr(p,and2a) by INCSTR; then
     InputVertices S = {x,b} by FACIRC_1:40;
   hence thesis;
 end;
```

```
theorem ::ThINCF2':
 for x,b being non pair set holds
  x ∈ InputVertices IncrementStr(x,b) &
  b ∈ InputVertices IncrementStr(x,b)
  proof
   let x,b be non pair set;
    InputVertices IncrementStr(x,b) = {x,b} by ThINCF2;
   hence thesis by ENUMSET1:9;
  end;

theorem ThINCW:
 for x,b being non pair set holds
  InputVertices IncrementStr(x,b) is without_pairs
  proof
   let x,b be non pair set;
    InputVertices IncrementStr(x,b) = {x,b} by ThINCF2;
   hence thesis;
  end;

:: 2's-BitComplementor

theorem ::ThBITCOMPIV:
 for x,b being non pair set holds
  InnerVertices BitCompStr(x,b) is Relation
   proof
    let x,b be non pair set;
    set p = <*x,b*>;
    set S1 = CompStr(x,b);
    set S2 = IncrementStr(x,b);
    set S  = BitCompStr(x,b);
A:   S = S1+·S2 by BITCOMPSTR;
     InnerVertices S1 is Relation & InnerVertices S2 is Relation
       by ThCOMPIV,ThINCIV; then
     InnerVertices (S1+·S2) is Relation by FACIRC_1:3;
   hence thesis by A;
  end;

theorem ThBITCOMPCA:
 for x,b being non pair set holds
   x ∈ the carrier of BitCompStr(x,b) &
   b ∈ the carrier of BitCompStr(x,b) &
   [<*x,b*>,xor2a] ∈ the carrier of BitCompStr(x,b) &
   [<*x,b*>,and2a] ∈ the carrier of BitCompStr(x,b)
  proof
   let x,b be non pair set;
   set p = <*x,b*>;
   set S1 = CompStr(x,b);
   set S2 = IncrementStr(x,b);
   set S  = BitCompStr(x,b);
A:   S = S1+·S2 by BITCOMPSTR;
     x ∈ the carrier of S1 & b ∈ the carrier of S1 &
     [p,xor2a] ∈ the carrier of S1 &
     x ∈ the carrier of S2 & b ∈ the carrier of S2 &
     [p,and2a] ∈ the carrier of S2 by ThCOMPCA,ThINCCA;
   hence thesis by A,FACIRC_1:20;
 end;

theorem ThBITCOMPCA':
 for x,b being non pair set holds the carrier of BitCompStr(x,b) =
   {x,b} U {[<*x,b*>,xor2a],[<*x,b*>,and2a]}
  proof
   let x,b be non pair set;
   set p = <*x,b*>;
   set S1 = CompStr(x,b);
   set S2 = IncrementStr(x,b);
   set S  = BitCompStr(x,b);
```

```
E1: the carrier of S1 = {x,b} U {[p,xor2a]} &
    the carrier of S2 = {x,b} U {[p,and2a]} by ThCOMPCA',ThINCCA';
    the carrier of S = the carrier of (S1+·S2) by BITCOMPSTR
     .= (the carrier of S1) U (the carrier of S2) by CIRCCOMB:def 2
     .= ({x,b} U {[p,xor2a]}) U ({x,b} U {[p,and2a]}) by E1
     .= {x,b} U {[p,xor2a]} U ({x,b} U {[p,and2a]})
     .= {x,b} U ({x,b} U {[p,xor2a]}) U {[p,and2a]} by BOOLE:64
     .= ({x,b} U {x,b}) U {[p,xor2a]} U {[p,and2a]} by BOOLE:64
     .= {x,b} U {[p,xor2a]} U {[p,and2a]} by BOOLE:62
     .= ({x,b} U {[p,xor2a]}) U {[p,and2a]}
     .= {x,b} U ({[p,xor2a]} U {[p,and2a]}) by BOOLE:64
     .= {x,b} U {[p,xor2a],[p,and2a]} by ENUMSET1:41;
  hence thesis;
end;

theorem ThBITCOMPF1:
 for x,b being non pair set holds
   InnerVertices BitCompStr(x,b) = {[<*x,b*>,xor2a],[<*x,b*>,and2a]}
   proof
    let x,b be non pair set;
    set p = <*x,b*>;
    set S1 = CompStr(x,b);
    set S2 = IncrementStr(x,b);
    set S  = BitCompStr(x,b);
A:   S = S1+·S2 by BITCOMPSTR;
B:   InnerVertices S1 = {[p,xor2a]} & InnerVertices S2 = {[p,and2a]}
       by ThCOMPF1,ThINCF1;
     InnerVertices S = InnerVertices (S1+·S2) by A
                    .= (InnerVertices S1) U InnerVertices S2 by FACIRC_1:27
                    .= {[p,xor2a]} U {[p,and2a]} by B
                    .= {[p,xor2a], [p,and2a]} by ENUMSET1:41;
  hence thesis;
 end;

theorem ThBITCOMPF1':
 for x,b being non pair set holds
   [<*x,b*>,xor2a] ∈ InnerVertices BitCompStr(x,b) &
   [<*x,b*>,and2a] ∈ InnerVertices BitCompStr(x,b)
  proof
   let x,b be non pair set;
   InnerVertices BitCompStr(x,b) =
     {[<*x,b*>,xor2a],[<*x,b*>,and2a]} by ThBITCOMPF1;
   hence thesis by ENUMSET1:9;
end;

theorem ThBITCOMPF2:
 for x,b being non pair set holds
  InputVertices BitCompStr(x,b) = {x,b}
  proof
    let x,b be non pair set;
    set p = <*x,b*>;
    set S1 = CompStr(x,b);
    set S2 = IncrementStr(x,b);
    set S  = BitCompStr(x,b);
A:   S = S1+·S2 by BITCOMPSTR;
B:   InputVertices S1 is without_pairs & InputVertices S2 is without_pairs
       by ThCOMPW,ThINCW;
C:   InnerVertices S1 is Relation & InnerVertices S2 is Relation
       by ThCOMPIV,ThINCIV;
D:   InputVertices S1 = {x,b} & InputVertices S2 = {x,b}
       by ThCOMPF2,ThINCF2;
     InputVertices S = (InputVertices S1) U (InputVertices S2)
                          by A,B,C,FACIRC_1:7
                    .= {x,b} U {x,b} by D
                    .= {x,b} by BOOLE:62;
  hence thesis;
```

```
 end;

theorem ThBITCOMPF2':
 for x,b being non pair set holds
  x ∈ InputVertices BitCompStr(x,b) &
  b ∈ InputVertices BitCompStr(x,b)
  proof
   let x,b be non pair set;
    InputVertices BitCompStr(x,b) = {x,b} by ThBITCOMPF2;
   hence thesis by ENUMSET1:9;
  end;

theorem ::ThBITCOMPW:
 for x,b being non pair set holds
  InputVertices BitCompStr(x,b) is without_pairs
  proof
   let x,b be non pair set;
    InputVertices BitCompStr(x,b) = {x,b} by ThBITCOMPF2;
   hence thesis;
  end;

::-------------------------------------------------------------------------
:: for s being State of BitCompCirc(x,b) holds (Following s) is stable
::-------------------------------------------------------------------------

theorem ThCOMPLem32':
 for x,b being non pair set for s being State of CompCirc(x,b) holds
  (Following s).CompOutput(x,b) = xor2a.<*s.x,s.b*> &
  (Following s).x = s.x & (Following s).b = s.b
  proof
   let x,b be non pair set;
   let s be State of CompCirc(x,b);
   set p = <*x,b*>;
   set S = CompStr(x,b);
   reconsider x' = x, b' = b as Vertex of S by ThCOMPCA;
    InputVertices S = {x,b} by ThCOMPF2; then
    x ∈ InputVertices S & b ∈ InputVertices S by ENUMSET1:9; then
F:  (Following s).x' = s.x & (Following s).b' = s.b by CIRCUIT2:def 5;
B:  InnerVertices S = the OperSymbols of S by FACIRC_1:37;
C:  dom s = the carrier of S by CIRCUIT1:4;
D:  x ∈ the carrier of S & b ∈ the carrier of S by ThCOMPCA;
E:  [p,xor2a] ∈ InnerVertices S by ThCOMPF1';
   thus (Following s).CompOutput(x,b)
       = (Following s).[p,xor2a] by COMPOUT
      .= xor2a.(s·p) by E,B,FACIRC_1:35
      .= xor2a.<*s.x,s.b*> by C,D,PreLem2';
   thus thesis by F;
 end;

theorem ::ThCOMPLem22':
 for x,b being non pair set for s being State of CompCirc(x,b)
  for a1,a2 being Element of BOOLEAN st a1 = s.x & a2 = s.b holds
   (Following s).CompOutput(x,b) = ¬a1 'xor' a2 &
   (Following s).x = a1 & (Following s).b = a2
  proof
   let x,b be non pair set;
   let s be State of CompCirc(x,b);
   let a1,a2 be Element of BOOLEAN; assume
A:  a1 = s.x & a2 = s.b;
   thus (Following s).CompOutput(x,b)
         = xor2a.<*s.x,s.b*> by ThCOMPLem32'
        .= ¬a1 'xor' a2 by A,XOR2A;
   thus thesis by A,ThCOMPLem32';
 end;

theorem ThCOMPLem32:
```

```
 for x,b being non pair set for s being State of BitCompCirc(x,b) holds
  (Following s).CompOutput(x,b) = xor2a.<*s.x,s.b*> &
  (Following s).x = s.x & (Following s).b = s.b
  proof
   let x,b be non pair set;
   let s be State of BitCompCirc(x,b);
   set p = <*x,b*>;
   set S = BitCompStr(x,b);
   reconsider x' = x, b' = b as Vertex of S by ThBITCOMPCA;
    x ∈ InputVertices S & b ∈ InputVertices S by ThBITCOMPF2'; then
F:  (Following s).x' = s.x & (Following s).b' = s.b by CIRCUIT2:def 5;
B:  InnerVertices S = the OperSymbols of S by FACIRC_1:37;
C:  dom s = the carrier of S by CIRCUIT1:4;
D:  x ∈ the carrier of S & b ∈ the carrier of S by ThBITCOMPCA;
E:  [p,xor2a] ∈ InnerVertices S by ThBITCOMPF1';
   thus (Following s).CompOutput(x,b)
       = (Following s).[p,xor2a] by COMPOUT
      .= xor2a.(s·p) by E,B,FACIRC_1:35
      .= xor2a.<*s.x,s.b*> by C,D,PreLem2';
   thus thesis by F;
  end;

theorem ThCOMPLem22:
 for x,b being non pair set for s being State of BitCompCirc(x,b)
  for a1,a2 being Element of BOOLEAN st a1 = s.x & a2 = s.b holds
   (Following s).CompOutput(x,b) = ¬a1 'xor' a2 &
   (Following s).x = a1 & (Following s).b = a2
  proof
   let x,b be non pair set;
   let s be State of BitCompCirc(x,b);
   let a1,a2 be Element of BOOLEAN; assume
A:  a1 = s.x & a2 = s.b;
   thus (Following s).CompOutput(x,b)
         = xor2a.<*s.x,s.b*> by ThCOMPLem32
        .= ¬a1 'xor' a2 by A,XOR2A;
   thus thesis by A,ThCOMPLem32;
 end;

theorem ThINCLem32':
 for x,b being non pair set for s being State of IncrementCirc(x,b) holds
  (Following s).IncrementOutput(x,b) = and2a.<*s.x,s.b*> &
  (Following s).x = s.x & (Following s).b = s.b
  proof
   let x,b be non pair set;
   let s be State of IncrementCirc(x,b);
   set p = <*x,b*>;
   set S = IncrementStr(x,b);
   reconsider x' = x, b' = b as Vertex of S by ThINCCA;
    InputVertices S = {x,b} by ThINCF2; then
    x ∈ InputVertices S & b ∈ InputVertices S by ENUMSET1:9; then
F:  (Following s).x' = s.x & (Following s).b' = s.b by CIRCUIT2:def 5;
B:  InnerVertices S = the OperSymbols of S by FACIRC_1:37;
C:  dom s = the carrier of S by CIRCUIT1:4;
D:  x ∈ the carrier of S & b ∈ the carrier of S by ThINCCA;
E:  [p,and2a] ∈ InnerVertices S by ThINCF1';
   thus (Following s).IncrementOutput(x,b)
       = (Following s).[p,and2a] by INCOUT
      .= and2a.(s·p) by E,B,FACIRC_1:35
      .= and2a.<*s.x,s.b*> by C,D,PreLem2';
   thus thesis by F;
 end;

theorem ::ThINCLem22':
 for x,b being non pair set for s being State of IncrementCirc(x,b)
  for a1,a2 being Element of BOOLEAN st a1 = s.x & a2 = s.b holds
   (Following s).IncrementOutput(x,b) = ¬a1 '&' a2 &
```

```
   (Following s).x = a1 & (Following s).b = a2
  proof
   let x,b be non pair set;
   let s be State of IncrementCirc(x,b);
   let a1,a2 be Element of BOOLEAN; assume
A:  a1 = s.x & a2 = s.b;
   thus (Following s).IncrementOutput(x,b)
        = and2a.<*s.x,s.b*> by ThINCLem32'
       .= ¬a1 '&' a2 by A,AND2A;
   thus thesis by A,ThINCLem32';
 end;

theorem ThINCLem32:
 for x,b being non pair set for s being State of BitCompCirc(x,b) holds
  (Following s).IncrementOutput(x,b) = and2a.<*s.x,s.b*> &
  (Following s).x = s.x & (Following s).b = s.b
  proof
   let x,b be non pair set;
   let s be State of BitCompCirc(x,b);
   set p = <*x,b*>;
   set S = BitCompStr(x,b);
   reconsider x' = x, b' = b as Vertex of S by ThBITCOMPCA;
    InputVertices S = {x,b} by ThBITCOMPF2; then
    x ∈ InputVertices S & b ∈ InputVertices S by ENUMSET1:9; then
F:  (Following s).x' = s.x & (Following s).b' = s.b by CIRCUIT2:def 5;
B:  InnerVertices S = the OperSymbols of S by FACIRC_1:37;
C:  dom s = the carrier of S by CIRCUIT1:4;
D:  x ∈ the carrier of S & b ∈ the carrier of S by ThBITCOMPCA;
E:  [p,and2a] ∈ InnerVertices S by ThBITCOMPF1';
   thus (Following s).IncrementOutput(x,b)
       = (Following s).[p,and2a] by INCOUT
      .= and2a.(s·p) by E,B,FACIRC_1:35
      .= and2a.<*s.x,s.b*> by C,D,PreLem2';
   thus thesis by F;
 end;

theorem ThINCLem22:
 for x,b being non pair set for s being State of BitCompCirc(x,b)
  for a1,a2 being Element of BOOLEAN st a1 = s.x & a2 = s.b holds
   (Following s).IncrementOutput(x,b) = ¬a1 '&' a2 &
   (Following s).x = a1 & (Following s).b = a2
  proof
   let x,b be non pair set;
   let s be State of BitCompCirc(x,b);
   let a1,a2 be Element of BOOLEAN; assume
A:  a1 = s.x & a2 = s.b;
   thus (Following s).IncrementOutput(x,b)
        = and2a.<*s.x,s.b*> by ThINCLem32
       .= ¬a1 '&' a2 by A,AND2A;
   thus thesis by A,ThINCLem32;
 end;

theorem ThBITCOMPLem32:
 for x,b being non pair set for s being State of BitCompCirc(x,b) holds
  (Following s).CompOutput(x,b) = xor2a.<*s.x,s.b*> &
  (Following s).IncrementOutput(x,b) = and2a.<*s.x,s.b*> &
  (Following s).x = s.x & (Following s).b = s.b
 by ThCOMPLem32,ThINCLem32;

theorem ::ThBITCOMPLem22:
 for x,b being non pair set for s being State of BitCompCirc(x,b)
  for a1,a2 being Element of BOOLEAN st a1 = s.x & a2 = s.b holds
  (Following s).CompOutput(x,b) = ¬a1 'xor' a2 &
  (Following s).IncrementOutput(x,b) = ¬a1 '&' a2 &
  (Following s).x = a1 & (Following s).b = a2
 by ThCOMPLem22,ThINCLem22;
```

```
theorem ::ThCLA2F3:
 for x,b being non pair set for s being State of BitCompCirc(x,b) holds
   (Following s) is stable
  proof
   let x,b be non pair set;
    set p = <*x,b*>;
    set CompOut = CompOutput(x,b);
    set IncOut  = IncrementOutput(x,b);
    set S = BitCompStr(x,b), A = BitCompCirc(x,b);
   let s be State of A; set s1 = Following s, s2 = Following s1;
A:  dom s1 = the carrier of S & dom s2 = the carrier of S by CIRCUIT1:4;
B:  the carrier of S = {x,b} U {[p,xor2a],[p,and2a]} by ThBITCOMPCA';
    now let a be set; assume a ∈ the carrier of S; then
      a ∈ {x,b} or a ∈ {[p,xor2a],[p,and2a]} by B,BOOLE:8; then
C:    a = x or a = b or a = [p,xor2a] or a = [p,and2a] by ENUMSET1:8;
D:    s2.x = s1.x & s1.x = s.x & s2.b = s1.b & s1.b = s.b by ThBITCOMPLem32;
E1:   s1.[p,xor2a] = s1.CompOutput(x,b) by COMPOUT
                   .= xor2a.<*s.x, s.b*> by ThCOMPLem32;
E2:   s1.[p,and2a] = s1.IncrementOutput(x,b) by INCOUT
                   .= and2a.<*s.x, s.b*> by ThINCLem32;
E3:   s2.[p,xor2a] = s2.CompOutput(x,b) by COMPOUT
                   .= xor2a.<*s1.x, s1.b*> by ThCOMPLem32;
      s2.[p,and2a] = s2.IncrementOutput(x,b) by INCOUT
                   .= and2a.<*s1.x, s1.b*> by ThINCLem32;
     hence s2.a = s1.a by C,D,E1,E2,E3;
    end;
   hence Following s = Following Following s by A,FUNCT_1:9;
  end;
```