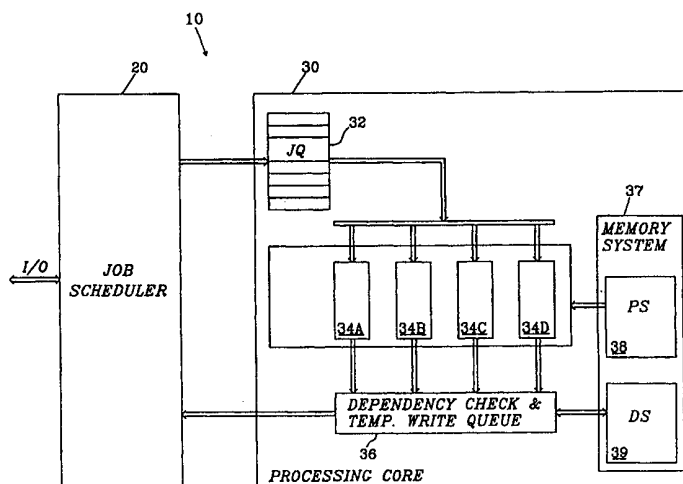




INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁷ : G06F 9/38, 15/16	A1	(11) International Publication Number: WO 00/29940 (43) International Publication Date: 25 May 2000 (25.05.00)
(21) International Application Number: PCT/SE99/02061 (22) International Filing Date: 12 November 1999 (12.11.99) (30) Priority Data: 9803901-9 16 November 1998 (16.11.98) SE 9901145-4 29 March 1999 (29.03.99) SE (71) Applicant: TELEFONAKTIEBOLAGET LM ERICSSON [SE/SE]; S-126 25 Stockholm (SE). (72) Inventors: HOLMBERG, Per, Anders; Flintbacken 18, S-118 42 Stockholm (SE). LINNEMARK, Nils, Ola; Tuvvägen 11, 3 tr, S-136 39 Haninge (SE). STRÖMBERGSON, Karl, Oscar, Joachim; Axgatan 6, S-431 40 Mölndal (SE). EGELAND, Terje; Björkviks Alléväg 2, S-139 56 Värmdö (SE). CARLSSON, Magnus; Lilla Liveredsvägen 11, S-428 33 Källered (SE). (74) Agents: HEDMAN, Anders et al.; Aros Patent AB, P.O. Box 1544, S-751 45 Uppsala (SE).	(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG). Published <i>With international search report.</i>	

(54) Title: MULTIPLE JOB SIGNALS PER PROCESSING UNIT IN A MULTIPROCESSING SYSTEM



(57) Abstract

The invention is directed towards a multiprocessing system (10) having multiple processing units (34A-D). According to the invention, for at least one of the processing units in the multiprocessing system, a first job signal is assigned to the processing unit for speculative execution of a corresponding first job, and a further job signal is assigned to the processing unit for speculative execution of a corresponding further job. The speculative execution of said further job is initiated when the processing unit has completed execution of the first job. If desirable, even more job signals may be assigned to the processing unit for speculative execution. In this way, multiple job signals are assigned to the processing units (34A-D) of the processing system, and the processing units are allowed to execute a plurality of jobs speculatively while waiting for commit priority. By assigning multiple job signals for speculative execution by one or more processing units, the effects of variations in execution time between jobs are neutralized, and the overall performance of the processing system is substantially improved.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakhstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

MULTIPLE JOB SIGNALS PER PROCESSING UNIT IN A MULTIPROCESSING SYSTEM

TECHNICAL FIELD OF THE INVENTION

5

The present invention generally relates to processing systems, and more particularly to a processing system having multiple processing units, and a method for handling job signals in such a processing system.

10

BACKGROUND OF THE INVENTION

Many conventional central processing systems, such as the APZ processing system in the known AXE Digital Switching System from Telefonaktiebolaget LM Ericsson, are built around a single processing unit, referred to as an execution pipeline in the AXE system. However, central processing systems based on a single processing unit have limitations with regard to capacity.

One way of increasing the processing capacity is to build the processing system as a multiprocessor system, i.e. a processing system with multiple processors or processing units operating in parallel. In conventional multiprocessing systems, each processing unit processes an input signal to execute a corresponding sequence of instructions, one input signal being assigned to each processing unit at a time.

In a so-called superscalar processor, which explores fine grained parallelism found between neighboring instructions, the functional units within the processor are arranged to simultaneously execute several instructions in parallel.

However, there is still a general demand for even more efficient processing systems.

SUMMARY OF THE INVENTION

The present invention constitutes a further development of the prior art multiprocessing system.

5

The invention is directed towards a multiprocessing system adapted for speculative execution of jobs. In a multiprocessing system, the processing units independently execute different jobs in parallel. However, at all times during parallel execution of jobs in a processing system adapted for
10 speculative execution, only one processing unit has commit priority and is allowed to execute its current job non-speculatively, performing write back to the memory system and committing signal sendings. Jobs in the other processing units are executed speculatively and may be flushed if dependencies are detected.

15

During development of such a processing system, a particular problem was encountered: If the job with commit priority has a longer execution time than the other speculative jobs, the speculatively executing processing units have completed execution of their assigned jobs long before completion of the
20 commit job. The speculatively executed jobs then have to wait until the commit job has been executed to completion so that one of the speculatively executed jobs can get commit priority. This generally means that the speculatively executing processing units simply wait for commit priority and valuable execution time is wasted, severely degrading the performance of the processing
25 system.

Therefore, it is a general object of the present invention to provide a more flexible and efficient speculative multiprocessing system as well as a more efficient method for handling job signals in such a multiprocessing system.

30

It is another object of the invention to provide a job queue for use in a multiprocessing system.

These and other objects are met by the invention as defined by the accompanying patent claims.

5 The general idea according to the invention is based on assigning, for at least one processing unit in the multiprocessing system, a first job signal to the processing unit for speculative execution of a corresponding first job, assigning a further job signal to the processing unit for speculative execution of a corresponding further job, and initiating speculative execution of said further job when the processing unit has completed speculative execution of the first
10 job. If desirable, even more job signals may be assigned to the processing unit for speculative execution, execution of the corresponding jobs being initiated as soon as the processing unit has completed speculative execution of the previously assigned job. The processing unit with commit priority may also be assigned a further job signal, the execution of which is initiated as soon as the
15 commit job has been completed.

By assigning multiple job signals for speculative execution by the processing units, the effects of variations in execution time between jobs are neutralized or at least greatly reduced, and the overall performance of the processing
20 system is substantially improved since the processing units are allowed to execute a plurality of jobs speculatively while waiting for commit priority.

In general, the protocol needed for assigning job signals to the processing units, keeping track of the job signals and handling the commit priority is
25 managed by a job signal queue in combination with appropriate control software or hardware.

In order to identify a job signal, its corresponding job or the results thereof in the operation of the processing system, each job signal is associated with an
30 identifier. The identifier may be in the form of a pointer to the storage position of the corresponding job signal in the job signal queue. Alternatively, the identifier comprises a unit label representing the identity of the processing unit

to which the job signal is assigned, and a job-signal distinguishing label representing the identity the job signal is given in the processing unit.

In a processing system adapted for speculative execution, the results of speculatively executed jobs are temporarily stored in a write queue arrangement, waiting to be committed. According to the invention, when a speculatively executed job gets commit priority the results of that job are retrieved by means of the identifier.

10 The invention offers the following advantages:

- The overall performance of the processing system is substantially improved;
- Flexible and efficient speculative execution is provided; and
- The effects of variations in execution time between jobs are neutralized.

15

Other advantages offered by the present invention will be appreciated upon reading of the below description of the embodiments of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

20

The invention, together with further objects and advantages thereof, will be best understood by reference to the following description taken together with the accompanying drawings, in which:

25 Fig. 1 is a schematic flow diagram of a method for handling job signals according to the invention;

Fig. 2 is a schematic diagram of a processing system according to the invention;

30

Fig. 3 is a schematic diagram illustrating the interrelationship between the job signal queue, the processing units and a write queue arrangement according to a first preferred embodiment;

- 5 Fig. 4 is a schematic diagram illustrating the interrelationship between the job signal queue, the processing units and a write queue arrangement according to a second preferred embodiment; and

- 10 Fig. 5 is a schematic diagram illustrating the interrelationship between the job signal queue, the processing units and a write queue arrangement according to a third preferred embodiment.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

- 15 Throughout the drawings, the same reference characters will be used for corresponding or similar elements.

Fig. 1 is a schematic flow diagram of a method for handling job signals according to the invention. In a processing system having multiple
20 processing units for job signal processing, job signals are assigned to processing units for execution of corresponding jobs. In general, the following steps are performed for at least one of the processing units: In step 1, a first job signal is assigned to the processing unit for speculative execution of a corresponding first job. In the case of the processing unit with
25 commit priority, the job is executed non-speculatively. In step 2, a further job signal is assigned to the processing unit for speculative execution of a corresponding further job. In step 3, when the processing unit has completed execution of the first job, speculative execution of the further job is initiated. It is of course possible to repeat steps 2 and 3 for even further
30 job signals.

In this way, the processing units are allowed to execute a plurality of jobs speculatively while waiting for commit priority. In theory, the number of jobs assigned to the same processing unit at the same time is not limited.

- 5 It should be understood that each job signal is intimately interrelated to its corresponding job and the results of said job.

In the following, the invention will be described with reference to an example of a realization of a multiprocessing system adapted for speculative execution. It
10 should however be understood that the invention is not limited thereto.

Fig. 2 is a schematic diagram of a processing system according to the invention. The processing system 10 basically comprises a job scheduler 20 and a processing core 30. The processing core 30 is adapted for speculative
15 execution of jobs, and the processing units 34A-D of the processing core 30 are preferably in the form of specialized hardware such as execution pipelines, although standard off-the-shelf microprocessors also are feasible.

The job scheduler 20 receives job signals, which represents events, from
20 external units as well as from the processing core 30, and schedules the job signals for processing by the processing core 30. An example of a specific type of job scheduler is the signal-processing unit (SPU) in the known AXE Digital Switching Systems from Telefonaktiebolaget LM Ericsson.

25 The processing core 30 basically comprises a job signal queue 32, simply referred to as a job queue, a plurality of execution pipelines 34A-D, a combined dependency checking unit and temporary write queue 36 for handling data dependencies, and a memory system 37 divided into a program store 38 and a data store 39.

30

The job signals from the job scheduler 20 are buffered in the job queue 32, which has a number of storage positions for storing the job signals. Each job

- signal is stored together with additional information in a respective storage position of the job queue 32. In general, each job signal comprises a header and data. In addition to administrative information, the header normally includes a pointer to software code in the program store 38, and the data of the job signal includes input operands necessary for execution of the corresponding job. The data could be a signal message from an external unit such as a regional processor or another processor. A job may be defined as the instruction stream specified by the signal header, and the job starts with the reception of the job signal and ends by the calling of an end-job routine. It should however be noted that the job signal itself normally does not include any instructions. The job signal normally includes a pointer to instructions in the software code stored in the program store 38, and operands required in the execution of the instructions, generally making the job signal self-contained.
- Preferably, the execution pipelines 34A-D independently "fetch" job signals from different storage positions in the job queue 32 to independently execute different jobs in parallel. Whenever an execution pipeline is free to start executing a new job, the job queue 32 is examined to find an unallocated job signal, and the unallocated job signal is assigned to the pipeline. The job signal is then processed in the execution pipeline and the corresponding job is executed. In this particular example, four pipelines are operable for executing four different jobs in parallel, independently of each other. At all times during parallel job execution, only one job signal in the job queue 32 is in commit position, allowing the execution pipeline to which the job signal is assigned to commit the corresponding job, i.e. performing write-back to the memory system 37 and committing signal sendings. The jobs in the other execution pipelines are executed speculatively and may be flushed if a data dependency is detected by the dependency-checking unit 36.
- A general requirement for systems where the information flow is governed by protocols is that certain related events must be processed in the received order. This is the invariant of the system, no matter how the system is

implemented. The commit order between jobs is defined by the arrival to the processing core and will generally not be changed. However, in a processing system handling job signals of different priority levels, it may be useful to put a job signal of higher priority level before job signals of lower priority.

5

In general, each execution pipeline comprises circuitry for fetching instructions from the program store, decoding the instructions, executing the instructions and performing memory write back. An example of a specific execution pipeline that can be used by the invention is the pipeline in the Ericsson AXE Digital
10 Switching Systems.

15

The dependency-checking unit 36 is generally implemented by using one or more read buffers associated with the execution pipelines 34A-D. When a pipeline fetches data from the data store 39, the read address or addresses are buffered in the read buffer(s). When the execution pipeline with commit
priority performs write-back to the data store 39, the write address to the data store is compared to the read addresses in the read buffer(s) to see if there are data dependencies between the jobs. If data read by a speculatively executing
job is subsequently modified by the committed job, a data dependency exists
20 and the speculatively executed job has to be flushed and restarted. The flushed job can be restarted from the job queue 32. Job signals corresponding to jobs that have been committed are removed from the job queue 32, thereby allowing new job signals from the job scheduler 20 to be buffered in the job queue 32.

25

30

Data store modifications proposed by a speculatively executed job are logged in the temporary write queue 36, but not written into the data store 39 until the job gets commit priority. When a speculative job gets commit priority, the entries in the temporary write queue 36 that belong to the job in question are immediately written into the data store 39. However, it should be understood that a commit job in execution generally does not store its data in the write queue, but simply forwards the corresponding memory address to the

dependency-checking unit to enable dependency checking. Job signals generated by a job with commit priority are forwarded to the job scheduler 20, where they are scheduled for later processing or sent to an input/output device (not shown) to be distributed to an external unit.

5

An example of a specific type of dependency checking unit and temporary write queue is disclosed in the international patent application WO 88/02513.

During development and testing of the processing system of Fig. 2, a particular
10 problem was encountered. When the job with commit priority has a longer execution time than the speculative jobs, the speculatively executing processing units have completed execution of their assigned jobs long before completion of the commit job. The speculatively executed jobs then have to wait until the commit job has been executed to completion and finally
15 committed so that one of the speculatively executed jobs can get commit priority. This generally means that the speculatively executing processing units simply wait for commit priority and valuable execution time is wasted, severely degrading the performance of the processing system. Simulations for telecommunication applications have shown that the capacity of only two
20 equivalent single-pipe processing systems was obtained by a processing system having a multi-pipe of four equivalent pipelines; a ratio of 2:4 which is far from optimal.

Of course, the above reasoning also holds true for a speculative job that has a
25 longer execution time than other speculative jobs. When a speculative job with a long execution time gets commit priority, the other speculative jobs have already been executed to completion, and are simply waiting for commit priority.

30 According to the invention, multiple job signals are assigned to at least one of the execution pipelines, and speculative execution of a next job is initiated as soon as the previous job has been executed to completion. In this way, the

execution pipelines are allowed to execute a plurality of jobs speculatively while waiting for commit priority. By executing several jobs per pipeline, the effects of variations in execution time between jobs are neutralized or at least reduced, and the above single-pipe-to-multi-pipe ratio is substantially improved.

In general, the protocol needed for assigning job signals to the execution pipelines 34A-D, keeping track of jobs that have been speculatively executed to completion, successively assigning commit priority to the job signals, and removing job signals that have been committed is managed by the job signal queue 32 in combination with appropriate control software or hardware.

The job queue 32 is normally an ordinary queue with a number of storage positions. As an example, the job queue 32 may be implemented in a common memory by logically dividing a predetermined part of the memory into a number of storage positions, each of which having its own individual memory address. In general, each storage position in the job queue 32 is divided into a number of fields, for example as described in Table I below.

Table I

Field name	Description
Valid	If set, the storage position contains a valid job signal
Taken	If set, the job signal has been assigned to an execution pipeline
Finished	If set, the corresponding job has been executed to completion, but not committed
U _L & D _L	U _L represents the identity of the pipe handling the job signal, and D _L represents the identity the job signal is given in the pipe. The field is normally a concatenation of U _L and D _L
Signal	The signal header and data

The field 'Valid' is used for indicating whether a storage position contains a valid job signal or not. When a job signal is received from the scheduling unit

20, it is placed in the first free position of the job queue 32, i.e. a position in which the Valid flag is not set. The Valid flag is then set to indicate that the 'Signal' field now contains a valid job signal, and the position is occupied.

- 5 The field 'Taken' is used for indicating whether the job signal has been assigned to a pipeline or not. When a job signal in a storage position of the job queue 32 is assigned to an execution pipeline, the Taken flag for that position is set to indicate that the job signal has been assigned.
- 10 The field 'Finished' is used for indicating that a job has been speculatively executed to completion, and the field 'U_L & D_L' is used for indicating the identity of the pipe and the identity the job signal and/or the corresponding job is given in that pipe. When a pipeline has executed the end-job routine for a first job, the Finished flag is set to signal that the job has been executed to
- 15 completion and is ready to be committed. The pipeline is now ready to fetch a second speculative job signal. The first job is assigned the job id "1" of the pipe in question and the second job to be executed in the pipe is assigned the job id "2". If the pipe finishes the second job before it receives commit priority, a third speculative job may be executed, and so on. In general, a trade-off between the
- 20 number of jobs per pipe and complexity is made, so in practice the number of jobs per pipe is limited.

In a processing system adapted for speculative execution, jobs may generally be in the following states: Not_Started, Started and Finished. These states may

25 be encoded in the job queue 32 in different ways, and it should be understood that the fields Valid, Taken and Finished of Table I is merely an example of one way of encoding these job states.

Preferably, the job queue 32 is associated with a pointer that points out which

30 storage position in the job queue 32 that is in commit position. The job signal in commit position has commit priority, and the execution pipeline handling

this job signal is enabled to perform write-back to the memory system 37 and send job signals to the job scheduler 20.

Fig. 3 is a schematic diagram illustrating the interrelationship between the job signal queue, the processing units and a write queue arrangement according to a first preferred embodiment. In this particular example, it is assumed that there are four different pipelines 34A-D operating in parallel. Each pipeline is assigned a unique unit label (1-4). It is also assumed that each pipeline is capable of handling two jobs. Therefore, for reasons of clarity and simplicity, the job queue 32 is illustrated as having eight storage positions. Each storage position includes the fields V (Valid), T (Taken), F (Finished), U_L (Pipe id), D_L (Job id) and SIGNAL. In Fig. 3, the field 'U_L & D_L' is illustrated as two separate fields U_L and D_L rather than as a single concatenation field. Both realizations are possible.

Job start

An example of how to start a new job will now be described. First, an execution pipeline that is ready for execution requests a new job signal from the job queue 32. Preferably, the pipeline also indicates which identity the new job signal should be given in the pipe. Next, the job queue 32 is traversed by conventional means, and the Valid flags and the Taken flags of the storage positions in the job queue 32 are examined to find the next position which contains a valid unallocated job signal. A pointer to the storage position of the unallocated job signal is transferred to job queue pointer register (JQP) of the requesting execution pipeline. The pointer may be in the form of a memory address to the storage position. The pipeline in question automatically reads its JQP register for the pointer to the next job signal, and subsequently copies the job signal given by the pointer into the pipeline to start executing the corresponding job. The Taken flag for that position of the job queue 32 is set, and the identity of the pipeline and the identity the job is given in the pipe are written into U_L and D_L, respectively. In this way, the job signals stored in the

job queue 32 are assigned to the execution pipelines 34A-D, and the signal fields of the storage positions are set accordingly.

In the example of Fig. 3, each one of the storage positions of the job queue 32 contains a valid (V=1) job signal that has been assigned (T=1) to one of the execution pipelines 34A-D. The jobs corresponding to the job signals of the first four positions have been executed to completion (F=1), awaiting commit priority. The jobs corresponding to the job signals of the last four positions are in the process of being speculatively executed, but none of the jobs have been finished (F=0). As soon as a job has been speculatively executed to completion, the corresponding Finished field is set.

The results of the speculatively executed jobs are logged in a write queue arrangement 36, but not written into the data store 39 until the job is committed. According to the first preferred embodiment of the invention, as illustrated in Fig. 3, the write queue arrangement 36 includes a write queue that is common to all execution pipelines. Each entry in the write queue is then provided with a pipe-id tag and a job-id tag indicating which pipeline and which one of the speculative jobs of the indicated pipeline that the entry belongs to. The pipe-id tag and the job-id tag correspond to U_L and D_L , respectively. Accordingly, each entry in the write queue includes an address (ADDR) to the data store 39, associated data (DATA), a pipe-id tag (U_L), a job-id tag (D_L) and a valid flag (V).

The job signal in the commit position, indicated by the arrow pointing to the job queue 32 in Fig. 3, has commit priority, and the execution pipeline handling this job signal is enabled to write to the data store 39. When a job signal is moved into commit position, the unit label of that storage position is examined to determine which pipe is responsible for the job signal and that pipe starts to commit all write operations and signal sendings for the job. By using the unit label U_L and the job-signal distinguishing label D_L stored in the commit position of the job queue 32, the relevant results can be retrieved from

the write queue and written into the data store 39. When this has been done, the position is released by clearing the Valid flag and the job queue 32 is stepped by conventional means to move a new storage position into commit position, thus successively assigning commit priority to the job signals of the job queue 32. The scheduling unit 20 is informed that the job has been executed to completion and the job queue 32 is now ready to receive a new job signal from the scheduling unit.

Job restart at flush

Now, an example of how to handle a flush will be described briefly. At flush, when a data dependency is detected, execution of the job to be flushed is normally interrupted in the pipeline, the job is put into a Not-Started state again, and those entries in the write queue that belong to the flushed job are invalidated.

15

In more detail, the pipe-id tag (U_L) of the entry in the write queue 36 for which a dependency has been detected identifies the pipeline that handles the job to be flushed. If the job is executing, the pipeline normally gets an interrupt. The JQP register of the identified execution pipeline holds a pointer to the storage position of the job signal in the job queue 32. By using the unit label U_L and the distinguishing label D_L of the storage position identified by the job queue pointer, the corresponding entries in the write queue can be invalidated. The Finished flag and the Taken flag of the corresponding storage position in the job queue 32 are removed, and the job can then be restarted. If the fields U_L and D_L of the relevant storage position in the job queue 32 are cleared, the job signal may be assigned to any execution pipeline that is free to start execution, and associated with a new unit label and a new job-signal distinguishing label. Alternatively, the job signal corresponding to the flushed job is reassigned to the pipeline identified by the unit label associated with the job signal, and the job signal is associated with a new job-signal distinguishing label D_L .

A job to be flushed does not necessarily have to be interrupted if it is executing. However, in terms of performance, it is more advantageous to interrupt the job.

5 Fig. 4 is a schematic diagram illustrating the interrelationship between the job signal queue, the processing units and a write queue arrangement according to a second preferred embodiment. Fig. 4 is identical to Fig. 3 except for the write queue arrangement 36. In Fig. 4, the write queue arrangement 36 comprises a number of separate write queues WQ1-WQ4, one write queue for
10 each one of the execution pipelines 34A-D. By having a dedicated write queue for each pipeline, it is sufficient to provide each entry in each write queue with a job-id tag instead of a job-id tag as well as a pipe-id tag. Consequently, each entry in the write queues WQ1-WQ4 includes an address (A) to the data store 39, associated data (DATA), a job-id tag (D_L) and a valid flag (V). When a job
15 signal in the job queue 32 is moved into commit position, the unit label U_L of that storage position is examined to determine which pipeline that is responsible for the job signal. At the same time, the corresponding write queue is also identified. By using the job-signal distinguishing label D_L stored in the commit position, the results of the job to be committed can be retrieved from
20 the relevant write queue and written into the data store 39. Correspondingly, at flush of a job, the write queue dedicated to the pipeline that handles the job can be identified by means of the associated unit label U_L . Subsequently, by using the associated distinguishing label D_L , the relevant entry in the write queue is identified and the results therein invalidated.

25

In the realization of Fig. 4, all speculative jobs that are placed behind the flushed job in the commit order are preferably also flushed to create space in the write queue 36. The reason for this is that the new execution of the flushed job, or the execution of another job, may need more space in the write queue
30 than the flushed job. Therefore, the position of the flushed job in the job queue is determined by examining the pointer in the JQP register of the pipeline in question. Next, the job queue 32 is traversed to find all speculative jobs that

are "younger" than the flushed job. For each found job, by using the unit label U_L and job-signal distinguishing label D_L associated with the job, the pipeline handling the job can be found and execution interrupted, and those entries that belong to the flushed job in the relevant write queue can be invalidated.

5

The functions for handling the job queue, such as associating labels to the job signals, writing information into the job queue fields and successively assigning commit priority to the job signals, are preferably controlled by software, e.g. in the form of a micro code instruction program or an assembler
10 program. Other functions of the processing system, such as handling flushes and controlling the write queue, may also be implemented by software. It is of course possible to implement the above functions in hardware instead.

Fig. 5 is a schematic diagram illustrating the interrelationship between the job
15 signal queue, the processing units and a write queue arrangement according to a third preferred embodiment. Fig. 5 is similar to Fig. 3, except for the job queue and the write queue arrangement. In Fig. 5, the storage positions in job queue 32 do not generally have any identification fields such as U_L and D_L . Instead the job queue pointer (JQP) is used as the sole identifier associated
20 with each job signal. At commit and at flush of a job, the job queue pointer is used in a search procedure to identify which pipeline that is responsible for the job. Each entry in the write queue includes a data store address (ADDR), associated data (DATA), a job queue pointer tag (JQP) and a valid flag (V). At
25 commit, the results of a job may be retrieved by means of the job queue pointer tag. At flush, the results of a job may be identified by means of the job queue pointer tag, and invalidated.

Preferably however, the write queue of Fig. 5 is provided with a field for a unit label tag to more easily find which pipeline that is responsible for a job.

30

Although the invention mainly has been described from the viewpoint that the protocol for assigning job signals to the processing units, keeping track of the

job signals and managing the commit priority is performed by the job signal queue and its associated control functionality, it should be understood that it is also possible to manage this protocol from the execution pipeline side.

- 5 It should also be understood that the term "processing unit" is not limited to specialized hardware but also includes other types of processing units such as standard off-the-shelf microprocessors. In the case of standard microprocessors, dependency checking is normally implemented by instrumenting read and write instructions with extra code in the application
- 10 software. Preferably, support for flushing is implemented in the application software and performed by the operating system or the virtual machine. If application software written for a single-processor system is to be migrated to and reused in a standard multiprocessor environment, the application software can be transformed via recompilation or equivalent. For example, a
- 15 sequentially programmed application software may be automatically transformed by adding appropriate code for supporting speculative execution, including code for dependency checking and code for storing a copy of modified variables to enable proper flush or roll-back of a job, and then recompiling the software. The operating system or virtual machine of the
- 20 standard multiprocessor system may also be modified to give support for speculative execution. For example, if a dependency is detected when executing code for dependency checking, the control may be transferred to the operating system/virtual machine which flushes the relevant job.
- 25 The embodiments described above are merely given as examples, and it should be understood that the present invention is not limited thereto. Further modifications, changes and improvements that retain the basic underlying principles disclosed and claimed herein are within the scope and spirit of the invention.

CLAIMS

1. A method for handling job signals in a processing system having multiple processing units (34A-D) for job signal processing, said method comprising the
5 steps of:
 for at least one of said processing units (34A-D),
 assigning (1) a first job signal to the processing unit for speculative execution of a corresponding first job;
 assigning (2) a further job signal to the processing unit for speculative
10 execution of a corresponding further job; and
 initiating (3) speculative execution of said further job when the processing unit has completed execution of said first job.
2. A method for handling job signals according to claim 1 further comprising
15 the step of successively assigning commit priority to the job signals of the processing system, one job signal at a time according to a commit order, a job corresponding to a job signal having commit priority being enabled for non-speculative execution including write back to a memory system (37).
- 20 3. A method for handling job signals according to claim 2, further comprising the step of associating each job signal with an identifier (U_L & D_L ; JQP) that is unique for the job signal.
4. A method for handling job signals according to claim 3, further
25 comprising the steps of:
 temporarily storing, for each speculatively executed job, results of the job in such a manner that said results are retrievable by means of the associated identifier;
 retrieving, for the job signal with commit priority, the temporarily stored
30 results of the corresponding job by means of the associated identifier, and performing write back of the retrieved results to the memory system (37).

5. A method for handling job signals according to claim 4, further comprising the steps of:

identifying, when a dependency with the job having commit priority is detected for a speculative job, the temporarily stored results of said speculative
5 job by means of the associated identifier; and

invalidating said identified results, whereby said speculative job is flushed.

6. A method for handling job signals according to claim 5, further
10 comprising the step of flushing all speculative jobs that are placed behind said flushed job in the commit order.

7. A method for handling job signals according to claim 5, further comprising the steps of:

15 clearing the identifier associated with a flushed job;

reassigning the job signal corresponding to the flushed job to a processing unit for execution; and

associating said reassigned job signal with a new identifier.

20 8. A method for handling job signals according to claim 3, wherein said identifier is a pointer (JQP) to the job signal in a job signal queue.

9. A method for handling job signals according to claim 3, wherein said
25 identifier comprises a unit label (U_L) representing the identity of the processing unit to which said job signal is assigned and a job-signal distinguishing label (D_L) representing the identity the job signal is given in said processing unit.

10. A method for handling job signals according to claim 9, further comprising the steps of:

30 temporarily storing, for each speculatively executed job, results of the job in such a manner that said results are retrievable by means of the

associated job-signal distinguishing label (D_L) or the associated unit label (U_L) together with the job-signal distinguishing label (D_L);

retrieving, for the job signal with commit priority, the temporarily stored results of the corresponding job by means of the associated job-signal distinguishing label or the associated unit label together with the job-signal
5 distinguishing label, and performing write back of the retrieved results to the memory system (37).

11. A method for handling job signals according to claim 9, further
10 comprising the step of reassigning a speculatively executed job that has been flushed to the processing unit identified by the associated unit label to restart execution of the job.

12. A processing system having multiple processing units (34A-D) for
15 processing job signals, said processing system comprising:

for at least one of said processing units (34A-D),

means (32) for assigning a first job signal to the processing unit for speculative execution of a corresponding first job;

means (32) for assigning a further job signal to the processing unit for
20 speculative execution of a corresponding further job; and

means for initiating speculative execution of said further job when the processing unit has completed execution of said first job.

13. A processing system according to claim 12, further comprising means for
25 successively assigning commit priority to the job signals of the processing system, one job signal at a time according to a commit order, a job corresponding to a job signal having commit priority being enabled for non-speculative execution including write back to a memory system (37).

30 14. A processing system according to claim 13, further comprising means for associating each job signal with an identifier that is unique for the job signal.

15. A processing system according to claim 14, further comprising:

means (36) for temporarily storing, for each speculatively executed job, results of the job in such a manner that said results are retrievable by means of the associated identifier; and

5 means for retrieving, for the job signal with commit priority, the temporarily stored results of the corresponding job by means of the associated identifier, and performing write back of the retrieved results to the memory system (37).

10 16. A processing system according to claim 15, further comprising:

means for interrupting a speculative job when a data-dependency with the job having commit priority is detected;

means for identifying the temporarily stored results of said interrupted speculative job by means of the associated identifier; and

15 means for invalidating said identified results, whereby said job holding a data dependency is flushed.

17. A processing system according to claim 16, further comprising means for flushing all speculative jobs that are placed behind said flushed job in the
20 commit order.

18. A processing system according to claim 16, further comprising:

means for clearing the identifier associated with a flushed job;

means (32) for reassigning the job signal corresponding to the flushed

25 job to a processing unit for execution; and

means for associating said reassigned job signal with a new identifier.

19. A processing system according to claim 15 or 16, wherein said identifier is a pointer (JQP) to the job signal in said storing means.

30

20. A processing system according to claim 14, wherein said identifier comprises a unit label (UL) representing the identity of the processing unit to

which said job signal is assigned and a job-signal distinguishing label (D_L) representing the identity the job signal is given in said processing unit.

21. A processing system according to claim 20, further comprising:

5 means (36) for temporarily storing, for each speculatively executed job, results of the job in such a manner that said results are retrievable by means of the associated job-signal distinguishing label or the associated unit label together with the job-signal distinguishing label; and

10 means for retrieving, for the job signal with commit priority, the temporarily stored results of the corresponding job by means of the associated job-signal distinguishing label or the associated unit label together with the job-signal distinguishing label, and performing write back of the retrieved results to the memory system.

15 22. A processing system according to claim 20, further comprising:

 a number of write queues (WQ1-WQ4), one write queue dedicated to each processing unit for temporarily storing, for each speculatively executed job of the processing unit, results of the job in such a manner that said results are retrievable by means of the job-signal distinguishing label (D_L) associated
20 with the corresponding job signal;

 means for determining, for the job signal with commit priority, which processing unit the job signal is assigned to based on the unit label (U_L) associated with the job signal;

25 means for retrieving the results of the corresponding job from the write queue (WQ) dedicated to the determined processing unit; and

 means for performing write back of the retrieved results to a memory system (37).

23. A job signal queue (32) in a processing system having multiple processing
30 units (34A-D) for processing job signals assigned to said processing units, said job signal queue having a number of storage positions, each storage position comprising:

a signal field (SIGNAL) for storing a job signal;
a first identification field (U_L) for storing a first label representing the identity of the processing unit to which said job signal is assigned; and
a second identification field (D_L) for storing a second label representing the identity the job signal is given in the processing unit identified by said first label.

24. A job signal queue according to claim 23, wherein said first identification field (U_L) and said second identification field (D_L) are concatenated into a single field (U_L & D_L).

25. A job signal queue according to claim 23, wherein each storage position further comprises a field (F) for storing a flag that indicates whether the job corresponding to the job signal stored in said signal field has been speculatively executed to completion.

26. A job signal queue according to claim 23, wherein each storage position further comprises:

a valid field (V) for storing a flag that indicates whether the signal field includes a valid job signal; and

a field (T) for storing a flag that indicates whether the job signal stored in said signal field has been assigned to a processing unit.

1/5

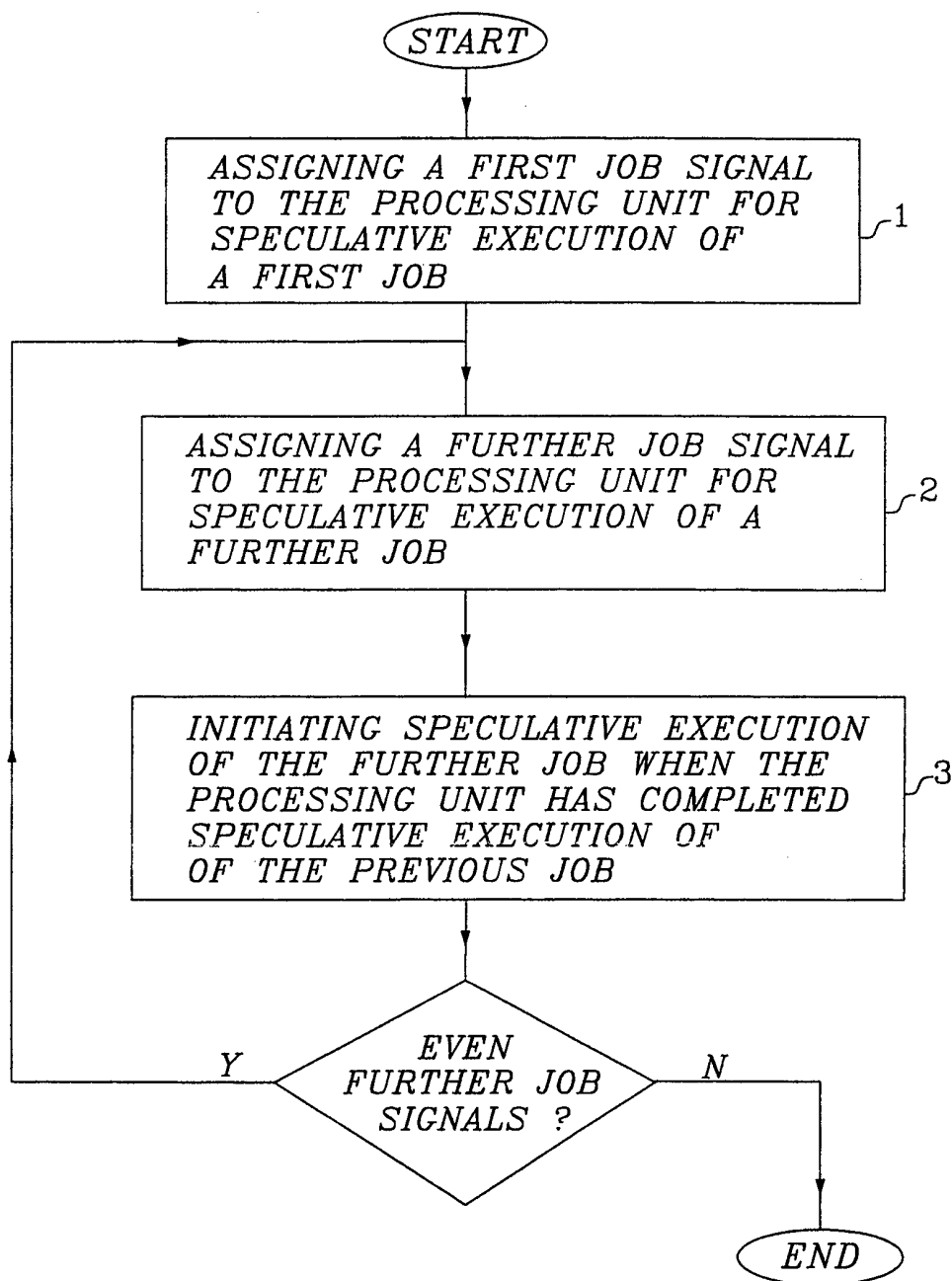


Fig. 1

2/5

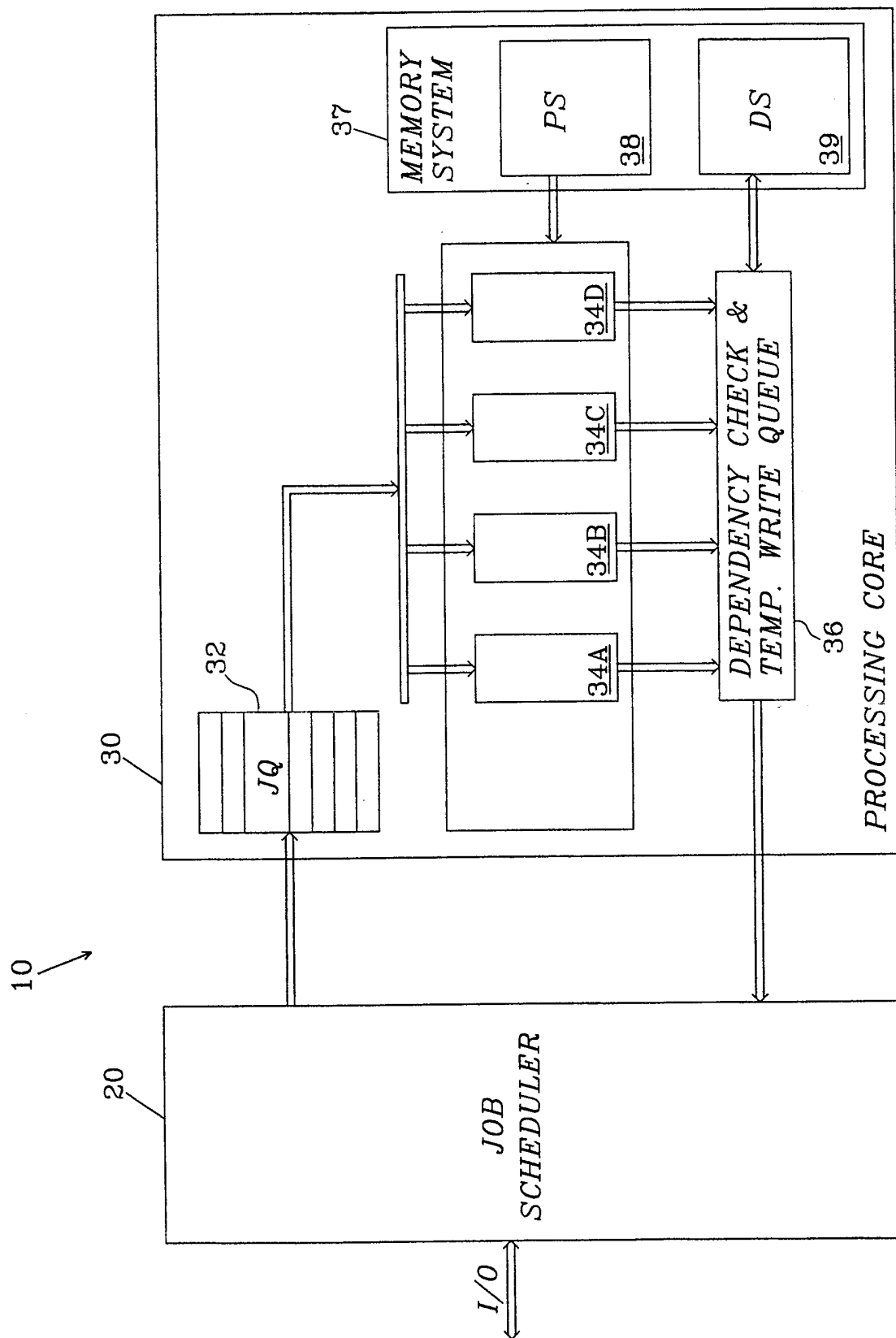


Fig. 2

3/5

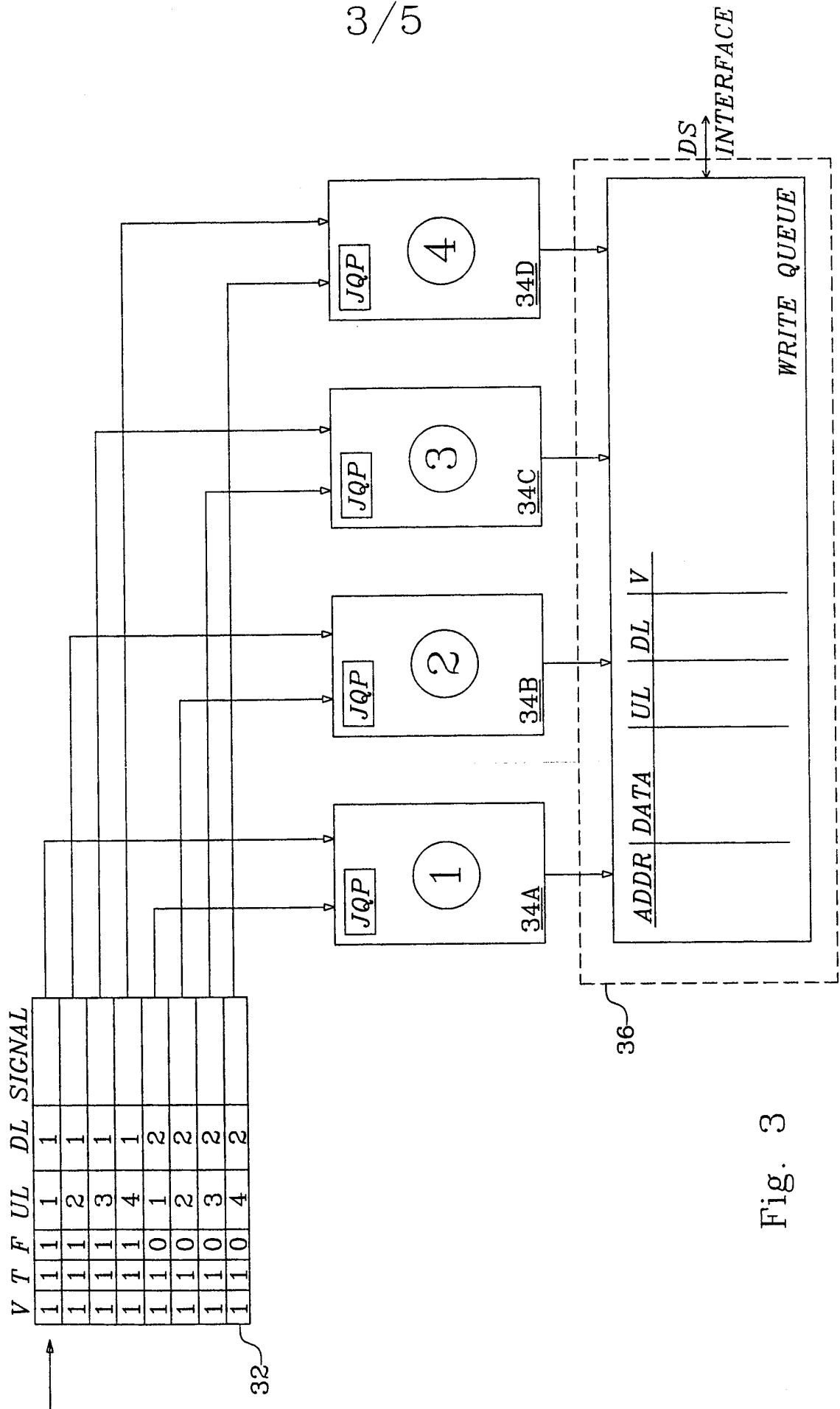


Fig. 3

4/5

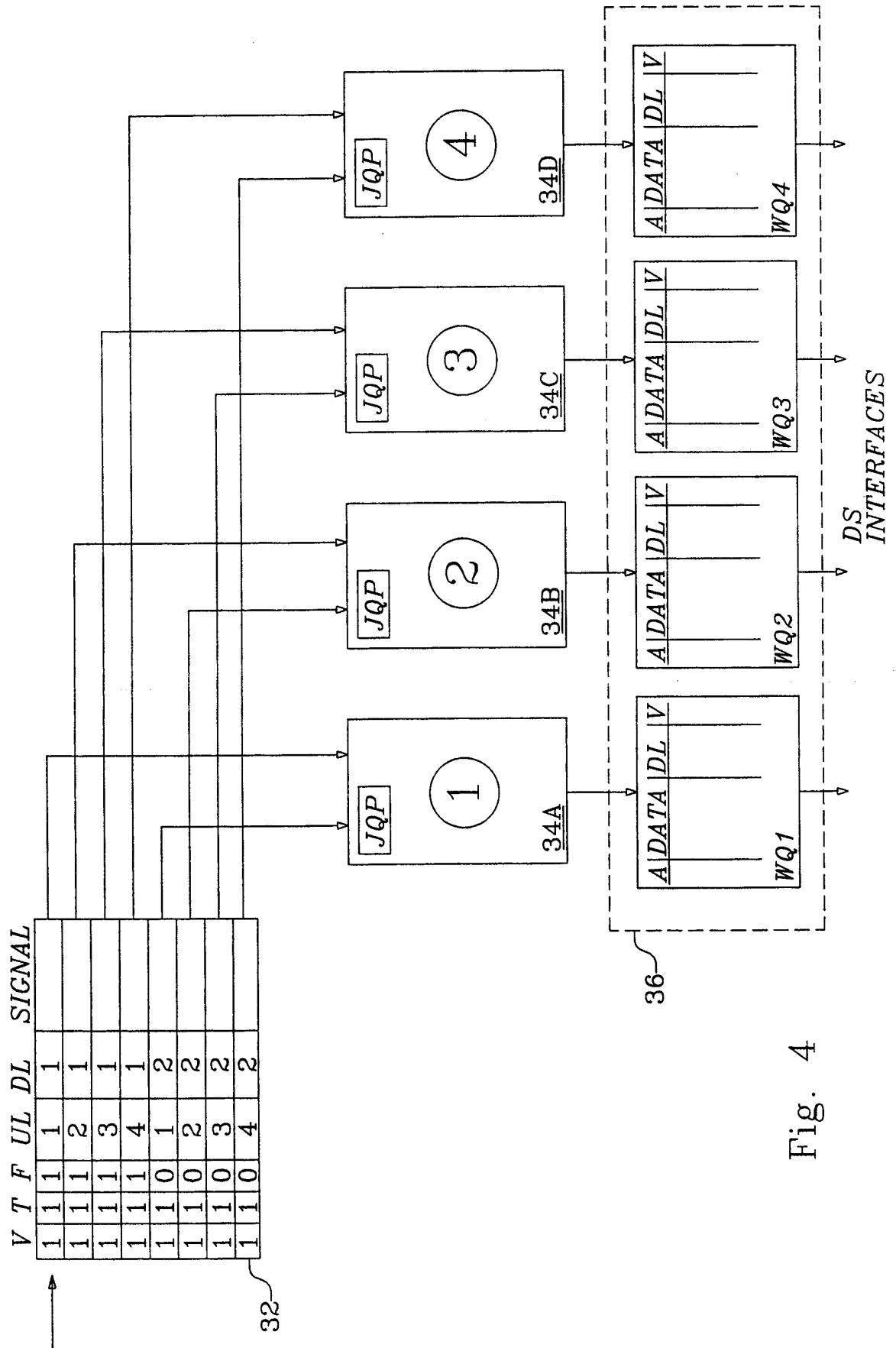


Fig. 4

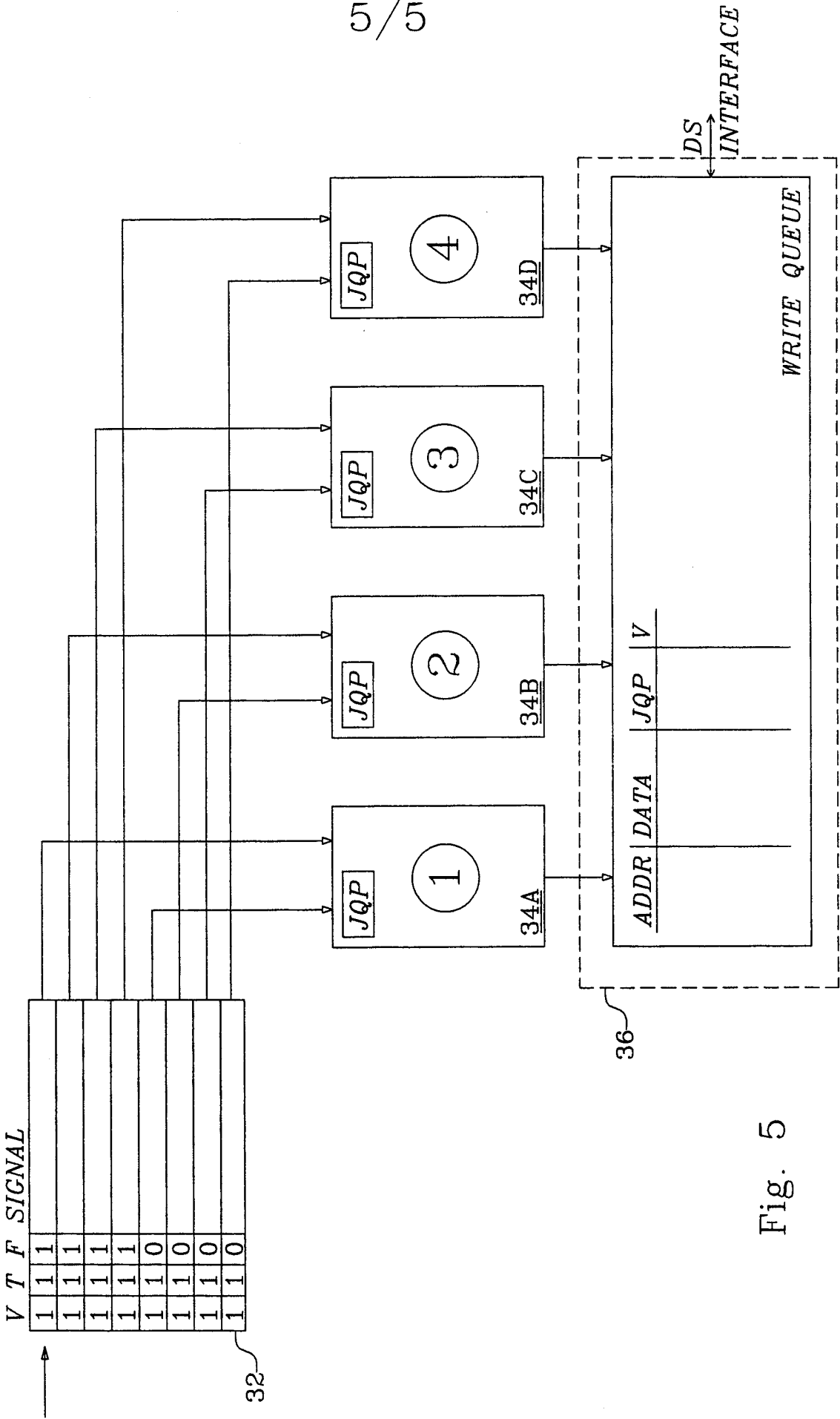


Fig. 5

INTERNATIONAL SEARCH REPORT

International application No.

PCT/SE 99/02061

A. CLASSIFICATION OF SUBJECT MATTER

IPC7: G06F 9/38, G06F 15/16

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC7: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5511172 A (KOZO KIMURA ET AL), 23 April 1996 (23.04.96), summary of the invention --	1,2,12,13
A	US 5072364 A (ROBERT L. JARDINE ET AL), 10 December 1991 (10.12.91), column 1, line 1 - column 2, line 68, claims 1-4 --	1-26
A	US 5287467 A (BARTHOLOMEW BLANER ET AL), 15 February 1994 (15.02.94), column 19, line 20 - column 28, line 68, claims 1-19 --	1-26

☒ Further documents are listed in the continuation of Box C.☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

17 February 2000

Date of mailing of the international search report

02-03-2000

Name and mailing address of the ISA:

Swedish Patent Office
Box 5055, S-102 42 STOCKHOLM
Facsimile No. +46 8 666 02 86

Authorized officer

Björn Edlund/CL
Telephone No. +46 8 782 25 00

INTERNATIONAL SEARCH REPORT

International application No.

PCT/SE 99/02061

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,A	US 5848257 A (RICHARD L. ANGLE ET AL), 8 December 1998 (08.12.98), column 5, line 39 - column 8, line 14 --	1-26
A	US 5812839 A (BRADLEY D. HOYT ET AL), 22 Sept 1998 (22.09.98), column 1, line 20 - column 2, line 58, claims 1-32 --	1-26
P,A	WO 9931589 A1 (INTEL CORPORATION), 24 June 1999 (24.06.99), claims 1-20, summary of the invention -- -----	1-26

INTERNATIONAL SEARCH REPORT

Information on patent family members

02/12/99

International application No.

PCT/SE 99/02061

Patent document cited in search report			Publication date	Patent family member(s)	Publication date
US	5511172	A	23/04/96	JP 5224927 A	03/09/93
US	5072364	A	10/12/91	AU 631874 B AU 5504790 A CA 2016254 A EP 0399760 A EP 0939364 A JP 2846406 B JP 3116235 A	10/12/92 29/11/90 23/11/90 28/11/90 01/09/99 13/01/99 17/05/91
US	5287467	A	15/02/94	DE 69229198 D EP 0509245 A,B JP 2051694 C JP 5143336 A JP 7085223 B US 5377336 A	00/00/00 21/10/92 10/05/96 11/06/93 13/09/95 27/12/94
US	5848257	A	08/12/98	NONE	
US	5812839	A	22/09/98	AT 184407 T DE 69420540 D EP 0661625 A,B SE 0661625 T3 SG 52391 A	15/09/99 00/00/00 05/07/95 28/09/98
WO	9931589	A1	24/06/99	AU 1911099 A	05/07/99